Generating Depth Maps from Stereo Image Pairs

Nicholas W. Walton

A thesis submitted for the degree of Doctor of Philosophy.
The University of Edinburgh.
April 2002

Abstract

In the 1960s a group of AI researchers came to the conclusion that automatic depth recovery by two cameras and a computer, the so-called Stereo Correspondence Problem (or *SCP*), was a simple task, and conquerable within an estimated five to ten years. Emboldened by the Information and Gestalt Theories, which say that the task is at least tractable, the field of Machine Vision was born. Forty years later, the task is still far from being solved, and is still a very active field of study.

This thesis summarises the developments which have taken place in stereo vision research since its inception. The various solutions which have been developed are described and their deficiencies noted.

An approach to generating depth maps from stereo image pairs using flow networks is proposed. This technique is suitable for implementation on low cost computing hardware and is intended to provide assistance to a human operator.

Results are presented which show the operation of this system on synthetic and natural images and the observed performance is analysed and discussed.

Declaration of originality

I hereby declare that the research recorded in this thesis and the thesis itself was composed and originated entirely by myself in the Department of Electronics and Electrical Engineering at The University of Edinburgh.

Nicholas Walton

Acknowledgements

I would like to thank, most of all John Hannah, my director of studies who has helped and guided me through a very long, and very difficult set of years, with equal quantities of patience, innovative concepts and butt-kicking. I'd also like to acknowledge the help, and guidance of everyone else in the Electrical Engineering Department for making the PhD years enjoyable and always interesting. Most notably, I'd like to thank George Taylor for his excellent tools and utilities to make our lives easier, and Shedden Masupe for sitting next to me, offering help and assistance, and always being there when I needed him.

Contents

List of figures

List of tables

Acronyms and abbreviations

1D	One-Dimensional
2D	Two-Dimensional
3D	Three-Dimensional
AI	Artificial Intelligence
CCD	Charged Coupled Device
CMOS	Complementary Metal-Oxide Semiconductor
DOG	Differential of a Gaussian
DP	Dynamic Programming
IEE	Institute of Electrical Engineers
IEEE	Institute of Electrical and Electronic Engineers
Kb	Kilobytes
LRH	Left-Right Heuristic
LOG	Laplacian of a Gaussian
Mb	Megabytes
SCP	Stereo Correspondence Problem

Chapter 1
Introduction

1.1 Aims and Objectives

A major portion of the research efforts of the computer vision community has been directed towards the study of the three-dimensional structure of objects using machine analysis of images[4–6]. Depth from multiple views is an important method, partly motivated by the human binocular vision system[7].

This thesis describes work done within this ongoing field; identifying, constructing and testing a novel binocular stereo algorithm.

There have been many attempts at solving the so-called Stereo Correspondence Problem, however the "general stereo vision problem" is far from being solved[8, 9]. There are some facts which give hope to the vision researcher, however.

Firstly, gestalt theory affirms that a pictorial scene contains all the information needed to perceive it[10], for instance, the human vision system can easily acquire the depth information of an object point from two perspective image planes.[9, 11]. There is a large body of psychophysical literature documenting various aspects of the human stereoscopic depth perception[11].

Secondly, animals which have far smaller brains than humans have the facility for stereo vision. Some lizards, such as chameleons, for example, have a far more flexible vision system whereby they can use their eyes together in a 'joint stereo' mode, or separately in 'independent mono' modes, something humans cannot.

Lastly, systems which can solve limited cases of the SCP have been produced many times[12, 13], utilising a plethora of different tools and modules, not necessarily mimicking the human stereopsis system, although some claim to. A great deal of literature exists on this subject [14].

In this document, some of these other methods will be examined, and where possible, compared and contrasted. Many of the solutions use pre-designed 'modules', such as edge-detection. Each of the major modules available to the designer will be discussed, highlighting their strengths, i.e. which problems they address, and weaknesses, i.e. the extra problems they pose while attempting to solve others.

The goal of this project is an aid to the human operator. The intended result is some measure of depth for objects in a scene, being viewed by a pair of stereoscopically mounted cameras. This system would signal the operator if some condition is met, for example a proximity warning. The aim is not to replace the eyes of the operator completely. However, with sufficient refining, it is possible that the system could be as accurate as a human for estimating distance. Once this aim is achieved, further 'back-ends' could be built onto such a system for more complex tasks.

The algorithm is aimed at low-end computer systems, and must therefore make the most use of explicit 'clues' within the images. In order to better understand the choices made in the design process, one needs to examine the history of the field, some of the findings of other researchers, and the lessons learned.

1.2 Background

1.2.1 Stereo Vision

Stereo Vision is an important passive method for extracting 3-D structure of a scene [3–7, 15–27].

Despite the flexibility, and lower complexity of passive stereo systems over active systems[19], binocular stereopsis is still limited by the amount of computation and hardware implementation difficulties [28, 29]. For this reason, laser, sonar and single-camera vision have proven more popular than stereo vision for use with tasks such as robotic vehicles [30, 31], because they use other methods to side-step the Stereo Correspondence Problem[1], which will be described below. Vision would be more widely adopted in Robotics if passive processes worked reliably all of the time[32]. Until recently, processing-intensive vision sensing has been impractical for mobile robotic use, due to the fact that the processing time was too long, or the cost was prohibitive.[33]

The act of recovering Stereo from passive imaging is generally reduced into the following three stages[1, 4, 18, 19, 34–37]:

- **Capturing and preprocessing the images,**

- **Recovering the disparity between the images** and
- **Using Trigonometry to recover the depths of the objects in the scene.**

It is the second stage, i.e. that of recovering the disparity, that forms the heart of the problem of stereo vision[1, 19, 22, 38, 39].

1.2.2 The Stereo Correspondence Problem

The aim of stereo vision research is to obtain depth information from a pair of camera images[40–42].

The task of stereo vision, can be posited thus:

> With a pair of eyes, each eye receives a slightly different image of the world due to its distinct position[6]. The human visual system can acquire depth using its two views[9, 11]. Stereo Vision systems attempt to imitate the depth extraction ability of the human visual system with the use of two cameras and a computer. [6, 43]. *If* we can uniquely determine which image segment in the first camera corresponds with which in the second camera[7, 44, 45], and *if* we know the geometry of the camera arrangement, and the properties of the cameras and their lenses, it is possible to calculate the distance to the original object (*back-projection*)[46].

The main problem is solving the accurate computation of the disparity between the left and right images[7, 14, 25, 26, 44, 45, 47–52]. The acquisition of depth by finding two corresponding points in the two perspective image planes, in the manner described above, is a process called *stereo correspondence*[9, 11].

As can be seen, the definition of stereo correspondence requires the correspondence mechanism to solve a lot of *ifs*. Such a conditional definition concerns us, since it renders stereo correspondence ill-posed[53–55].

Because stereo correspondence is so ill-posed, depth estimation is not a simple task[9, 13, 56], and is often called *The Stereo Correspondence Problem.*

This correspondence problem arises from the problem of determining which points in both images correspond to the same 3-D scene point[1, 4, 7, 44, 45, 57–60]. Figure 1.1 shows the ambiguity inherent in the stereo matching process. For the matching of just four objects, there are sixteen potential candidates[1,61].

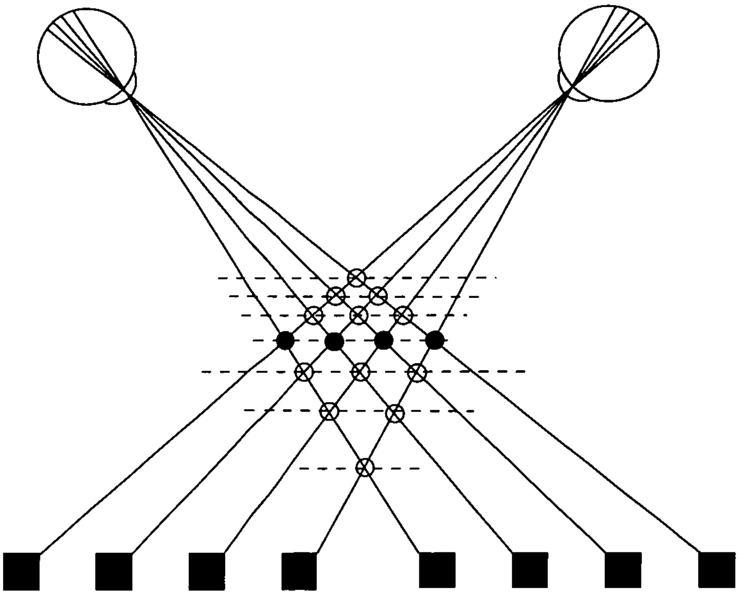


Figure 1.1: *Ambiguity in the correspondence between the two retinal projections. In this figure, each of the four points in one eye's view could match any of the the four in the other eye's view. Of the 16 possible matchings, only four are correct (filled circles), while the remaining 12 are "false targets" (open circles). [1, 2]*

The correspondence problem in inherently under-constrained, and therefore constraints have to be imposed by making assumptions regarding scene reflectance and structure[62]. Various constraints can be added to limit the amount by which the ill-posed SCP is a hindrance[20]. There are many such constraints, of which the most widely employed are the Photometric, Uniqueness, Coherence, Smoothness, Epipolar Constraints and Disparity Gradient. These constraints will be discussed further in Chapter Two. By constraining a system, the researcher can limit its requirements so that while it is still useful, it does not have to solve 'hard' problems like oddly shaped (discontinuous) objects, or transparent objects. Other research, such as that carried out by Walker, utilises Knowledge-Based Image Understanding (KBIU) models to solve problems previously addressed by 'loose' constraints[63].

In *binocular* stereo especially, the SCP is exacerbated by the restriction of using only two cameras. This is due to the lack of corroborating evidence with which to test matching hypotheses[58]. As will be discussed later, there are many other ways to obtain this corroborative data, but in general employ:-

- **more cameras** (violating the term binocular).
- **active scanning**, such as *ladar* (laser range-finding) or **steerable cameras** (violating the term passive).
- or **structured lighting**, which limits the system to a number of pre-configured scenes.

Of all these modifications, the addition of more cameras is considered an acceptable solution, since it does not involve any additional active components, and does not restrict the system to a smaller set of situations, in the way that structured lighting would.

Adding a third camera (tri-nocular system) can ameliorate some problems of correspondence[4, 8, 40, 64]. If one is willing to add an extra camera, the third camera can be used in one of two ways. It can be used to corroborate the depth estimation given by the other two cameras, or it can be paired alternately with one of the other two to gain a slightly different view of the scene. Some researchers (such as Kanade and Okutomi, who will be discussed in Chapters Two and Three), use many more cameras, so-called *multi-baseline* stereo to gain even more evidence for the Correspondence stage of the system.

Unlike many other fields of research, stereo vision researchers are presented with proof that their goal is achievable, every day, since they themselves contain a fully functional stereo system. It can be argued, however, that humans use much of their knowledge and experience to guide their perception of depth, since humans can describe the shape of objects they have never seen before [65], and can still 'estimate' depth with only one eye open, albeit only approximately. On the other hand, studies using random-dot stereograms show that humans can solve the SCP *without* recourse to prior knowledge[1, 29, 34, 36]. This, at least, gives hope that a knowledge-less depth estimator may be achievable.

Over the past twenty years, computational models for most of the 3-D 'clues' that appear to be used by the human visual system have been proposed, in fact many models for each system[66].

For most systems employing computational stereo, the goal is to obtain a *depth map*.

1.2.3 What is meant by 'Depth Map'?

Consider an artist who, rather than using a sheet of canvas, uses a piece of glass. In order to paint the scene in front of him, he simply dabs a dot of colour to match the object he sees behind the glass. If he keeps his head still, and uses a small enough brush, he will have a fair representation of the scene through the glass.

Consider, rather than painting the scene, there is an assistant who holds a tape measure up to the glass, and measures the distance to each object in the scene. The assistant calls out the distance, and the painter dabs a gray spot according to the distance. A black spot is dabbed for items that are the greatest distance, and a while spot for the closest, with varying grey levels for objects in-between. This is a depth map. Figure 1.2a shows an example of a depth map generated automatically from two views of a computer-generated corridor scene.

The term *depth map* occurs many times in the literature as does the term *disparity map* [3,4,8, 39,67–72]. There is a subtle difference in meaning, although they are often used interchangeably because depth information is included in the disparity[6]. A disparity map contains merely the pixel disparities regained from the SCP Solver, whereas a depth map goes one step further and uses inverse trigonometry to recover depth. The term '*Ground Truth*' is sometimes used interchangeably with both. When a point in one image, and another point in another image are considered to be projections of the same object in the scene, they are said to be *matched.* Each of the points has a Cartesian coordinate relative to the origin of the camera it occurs in (x_l, y_l) and (x_r, y_r), for a binocular rig. The difference in their relative positions (in pixels) is recorded in a 'cell' at location (x_l, y_l) in a disparity map. When the reverse trigonometric transform is applied to convert the disparity back into 3-D world coordinates, the depth is subsequently stored at location (x_l, y_l) in a depth map.

However, because there is a one-to-one relationship between pixel disparity and depth [1], given in Equation 1.1, the difference is purely semantic, and most methods halt at the disparity map stage, since the inverse trigonometry (back projection) is considered a trivial problem once the SCP has been solved.

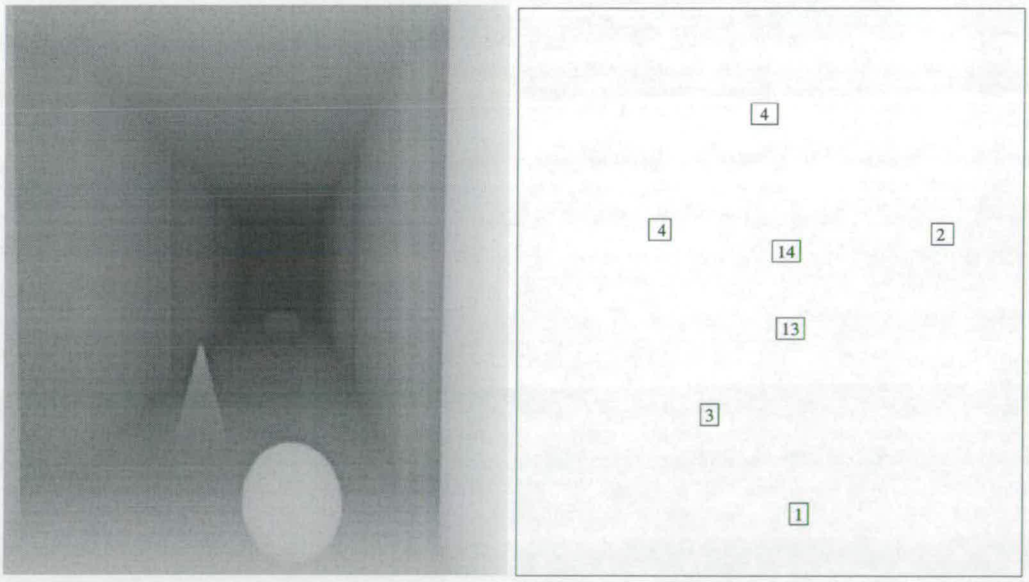
$$Depth = \frac{Baseline * FocalLength}{Disparity} \tag{1.1}$$

Depth maps can be calculated automatically from 3-D rendered scenes using a technique known as *Ray Tracing*. Ray Tracing works in the same way as the hypothetical painter above. A ray is

[1] if transparency and reflection are constrained out from consideration

'shot' from each pixel of the image into the computer-generated scene, and the pixel gains the colour of the object that the ray strikes. If this is modified slightly so that the pixel gains a grey-level colour proportional to the distance to the object that the ray struck, then this is identical to the assistant and tape measure example above. Figure 1.2 shows the result of performing this process on a synthetic 'corridor' scene.

Stereo Algorithms can produce *sparse* or *dense* depth maps. While *feature-based* matching can be implemented efficiently[73], it typically only produces a sparse depth map[19, 74]. Whereas although *area-based* is computationally expensive[73], it produces a dense depth map[68, 74], A sparse depth map arises when there are only a relatively small number of image feature-points where the correspondence can be solved (such as at object boundaries, i.e. gray-level discontinuities, or edges). Simple feature-matching, such as this, therefore produces an impoverished description of the scene as a whole[75]. In order to obtain a dense depth map from such feature-based approaches, some interpolation is needed. Many applications, such as View Synthesis and object grasping require a dense depth map[8], whereas tasks such as robotic obstacle avoidance may only require a token-based sparse-depth map.



a) A dense depth map b) A sparse depth map.

Figure 1.2: *Representing the depth in a scene using a dense and a sparse depth map*

In general it is much harder to produce a dense disparity map due to the large numbers of occluded areas, errors introduced by the cameras and by parallax, especially when area-based methods are used, which are susceptible to such problems [4, 8, 34, 64].

1.2.4 Occlusions and Parallax

Among the most notorious problems in stereo vision are repetitive patterns and textureless areas, and the presence of depth discontinuities and half-occluded regions[8]. Occlusions and textureless areas pose such problems because all stereo systems are given certain common-sense constraints to help their matching, and the presence of these artifacts forces the system to bend or break these constraints.

Occlusions occur when an object close to the camera obscures or *occludes* a more distant object. When viewed from different angles, the *occluding object* obscures different sections of the *occluded object*. In the most severe case, the occluding object will totally obscure the occluded object from one vantage point, but not from the other, presenting the matcher with an almost insoluble problem. Figure 1.3 shows a typical layout which would cause an occlusion event to occur.

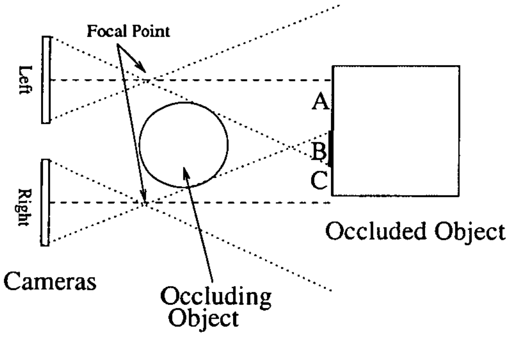


Figure 1.3: *Top view of a stereo camera arrangement observing a scene with occlusion*

In Figure 1.3, The surface of the occluded object is divided into three areas. Areas A and C are visible from only one of the two cameras. Section B, however is the only part of the surface which is visible from both. Since so much of the area of the occluded object is hidden in one or other stereoscopic view, the matcher will have problems reconstructing that area of the scene, and most algorithms will merely leave the depth map black at that point. Figure 1.4 shows the view from the cameras. If the occluded object is texture-rich, the areas which both cameras can observe can easily be identified, but in the absence of texture, such as in the Figure, correct matching is impossible, and most likely to cause incorrect results.. Work, such as that done recently by Zaki et al uses model-matching to attempt to reconstruct occluded objects[12]. This, however, requires *a priori* knowledge of the objects which are to be viewed.

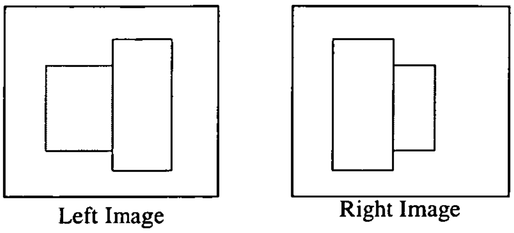


Figure 1.4: *The effects of occlusion on the visibility of the objects.*

Parallax is a problem which befalls area-based algorithms mainly, since they attempt to match two areas of a scene which may or may not be compressed or expanded along the horizontal axis, thanks to parallax. An extreme example of this is given in Figure 1.5. As can be seen the object is longer than the baseline, and the forward edge of it is close to the cameras. The result, which can be seen in Figure 1.6 is that the object appears *squashed* in the right camera view. The object has been given a striped texture to highlight the compression effect.

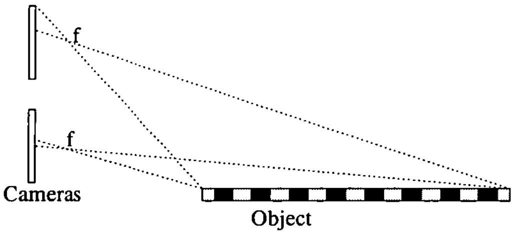


Figure 1.5: *Top view of a stereo camera arrangement observing a scene with extreme parallax*

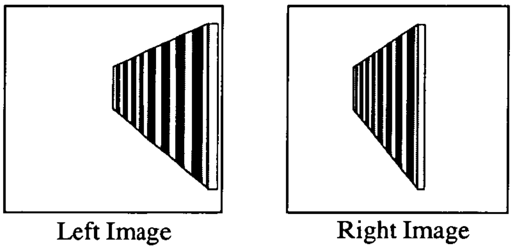


Figure 1.6: *The effects of parallax on the observed images of an object.*

If any stereo system is to become truly "general purpose", it must be able to cope with such problems as occlusion and parallax, as well as noisy cameras, non-uniformly lit environments and so on. Given enough resources, and enough time, such a system is likely to be built, however the services that such a system can offer are of benefit now, so therefore restrictions must be placed upon the generality of the solution, and the computing hardware which must be used to tackle it. Erken stated that even by 1999, "Industrial machine vision at this point still offers too little for too much to attract consumer interest"[76]

1.3 Restrictions

At MIT in 1967, the famous AI researcher, Marvin Minsky sent around an internal report with the message that it is an interesting practical and theoretical problem, putting bounds on how much computation one needs to find the stereo-disparity between two narrow-angle stereo scenes[77], in order to inspire the researchers there and prompt new work.

That statement expresses exactly the driving force behind this project, more than thirty years later. To design and implement a novel stereo algorithm with the goal to keep the processing requirements and the reliance on constraints to a minimum, and to fully test that algorithm with difficult data to see how versatile such a solution truly is.

Since the aim is to produce a low-cost high-speed stereo matching algorithm, there are a number of restrictions that must be placed on the intended solution.

If the system is to be low cost, then it must run on either off-the-shelf hardware, such as a standard PC or Power PC, or it must be easily converted into a low-cost chip design. In either case, it must not have excessive storage or processing power requirements.

The decision of what constitutes "large" quantities of memory or hard drive space is dependent upon the application. In either case, it must not use such a large quantity of memory or hard drive space that its provision would dominate the cost of the product. Therefore, as time passes, the parameters of a "large" system will grow with technology.

If the system is to be fast, then it must be as close to 'real-time' as possible, i.e., as little delay between image input and depth field output as possible.

By specifying a set of requirements by which the solution must abide, one restricts oneself to the choice of methods by which the solution can be achieved. For example, if unlimited set-up time is allowed, then a Neural Network is acceptable[78]. If the expected target hardware platform is a large multi-processor machine, and slow runtimes are acceptable then a frequential algorithm may be desirable. All of these algorithms have been inspected, and their relative merits given later. However all but a few have proven to be too costly to implement, both in terms of hardware, and of computation.

1.4 Applications

Why is such effort being channelled into this area of research? The simple answer is that there are many applications for a stereo-capable system[79]. As well as drawing much from the biological sciences, AI often produces theories and hypotheses which prove pertinent in return, and some of these hypotheses have subsequently been supported by evidence[80], the collected works of Marr, Poggio, Mayhew, Frisby and Grimson being good examples of this.

Depth Perception has long been recognised as necessary for a truly flexible robot vision system[59], and therefore much of computational stereo research focussed on automated navigation for mobile robots. However, the applications can be far more mundane than this. As well as aiding automata, stereo systems can also aid human operators. There are many situations where human depth sense can be augmented with these systems.

Consider the case of a human operating a machine, such as a car, where a stereo view of the rear section would be helpful for gauging distances[33,81] to prevent collisions[13,49], or perhaps follow a road or track[82]. in medicine, stereo vision has been put to use automatically analysing the outputs of MR, CT, X-Ray, NMR, ultrasound, PET and SPECT images [79].

As well as guidance, and medical tasks, stereo information can be of benefit in many other industries. Consider the animation or video games industries, in which much effort is put into digitising humans performing tasks for study or creation of more life-like animation. In general this is done by the use of phosphorescent dots attached to limbs, or using a 'stereo cage', i.e.. a cubic lattice of six or more cameras. The first requires less power, but results in no textual data, simply skeletal movements onto which polygonal 'bodies' must be mapped. The second method requires a large cage, and many cameras, which produces both textual and skeletal movement data, but is extremely expensive, involves a lot of processing, and also restricts the action to the dimensions of the cube. A stereo system involving two cameras has a much less restricted field of view, and is much less costly. Also, due to the lower number of cameras, it entails much less processing overhead.

It is hoped that through this brief introduction, the reader is able to understand some of the fundamental difficulties and dichotomies facing the stereo researcher. It is also hoped that the flow of reasoning through the thesis, and decisions taken during the project, is easier to follow.

1.5 Thesis Plan

The aim of this thesis is the design and production of a vision algorithm, aimed at creating a stereo vision system for low-cost hardware, and preferably for varied applications.

This project examines the thesis that a system to solve the Stereo Correspondence Problem can be created using the technology of *Flow Networks*. The hypothesis is that using this technology for the disparity recovery engine could produce a system to solve a generalised form of the SCP, and also be optimisable such that it could be implemented on low-cost or general purpose hardware.

The thesis will describe each step explaining the techniques, and comparing them against those employed by others to solve the SCP. The last chapter, as well as discussing the conclusion to the research will also put forward some ideas for future research.

Chapter Two contains a discussion of the field, and discusses other means by which stereo has been attempted. Chapter Two concludes with a section on the techniques (such as Directed Graphs) which were borrowed from other disciplines, and the standard techniques more commonly used in AI and Stereo Vision.

Chapter Three builds upon this and discusses some techniques which others have used, dividing them up into two categories, Spatial and Frequential approaches.

The approach discussed briefly at the end of Chapter Two is expanded and described in Chapters Four and Five. The preparation of the data and other low level tasks are described in Chapter Four, whereas the latter stages, i.e. the acquisition of the disparity, and ultimately, the depth map pair, are described in Chapter Five.

Chapter Six shows some experimental results, grouped by dataset, and Chapter Seven gives conclusions, and proposes further research.

Chapter 2
Stereo Vision and Artificial Intelligence

2.1 Introduction

Artificial Intelligence (AI) is the design and study of computer programs that behave intelligently. These programs are constructed to perform as would a human or an animal whose behaviour is considered intelligent[83].

Artificial Intelligence has only recently come to the attention of the general public due to a number of limited successes. It has, however, been an active field of research since the early 1960's. AI encompasses many diverse areas of study, which generally fall into the categories of Connectionism (i.e. neural networks and genetic algorithms), Natural Language Processing, Robotics and Computer Vision.

The challenge to AI is that humans do much of their "intelligent" selection inside non-introspectable subsystems[84], and must reinvent those specialised, parallel and partly analog subsystems with general digital hardware[83]. How the brain computes disparity, and thus achieves stereoscopic depth perception, has been the subject of many studies[11]

Frank Rosenblatt was determined to simulate the brain, and in 1957, based upon a number of observations he made about biological learning mechanisms, invented the "Perceptron Net"[85], a forerunner to modern-day Neural Nets. He demonstrated an optical pattern "learning machine" in 1960 based upon Perceptrons.

In a book published nine years after Rosenblatt's demonstration, "Perceptrons: An Introduction to Computational Geometry", Minsky and Papert discuss the usefulness of Perceptrons, and criticise their usefulness[86]. After their publication, interest in Networks declined. John Hopfield revived this interest in 1982, and this led to much research into the new "Hopfield Nets", and Neural Networks in general. One famous example of the application of a Hopfield Net was "NetTalk".

Despite the 'black-box' nature of the human brain, computers can already be programmed to perform certain tasks, such as chess playing, at an apparent skill level higher than all but a very small number of human experts[87].

On the topic of vision, since humans agree so closely on their depth impressions, it is fair to assume that their major assumptions are the same, and are therefore subject to identification and analysis[88]. In fact, in those places where the brain now does have an advantage, it would seem that this advantage may be short-lived[87].

As will be shown, the boundaries are not quite as clear-cut as this. In fact, part of the history of Computer Vision research is its swings between implementations of the flow of control[84]. In recent years, Knowledge-based and connectionist methods have been used to solve the SCP with varying degrees of success.

In order to highlight how views and theories on underlying implementation have changed,the progress towards a general solution to the problems of Computational Vision and Chess by AI will be compared, and also used to demonstrate why there has not yet been an effective solution to the General Stereo Vision Problem.

After the section on AI's history, the rest of the chapter is organised into three notional parts of the stereo vision pipeline. Firstly, the task of capturing images, is discussed. Then there is a discussion of the major algorithms and systems in use, along with their pros and cons. Lastly, a short section is devoted to the task of displaying the 3-D data once it has been recovered from the 2-D images.

2.2 History

2.2.1 Predictions of the Future

In the 1960s, computing technology developed sufficiently so that real-time processing of audio and video signals became a reality[65]. Thanks to the advances in motor, microphone and camera technology, the standard tasks that are usually considered as simple functions within humans were deemed to be the simplest, and therefore the most likely to be the first AI problems solved. These problems covered such day-to-day tasks as walking, hearing and seeing.

The AI experts believed, however, that the more cognitive 'frontal-lobe' activities, such as chess playing, natural language processing and speech would be the hardest. In his book, "Vision", another renowned AI researcher David Marr, reports that in the sixties, the problem of stereo vision was tipped for completion, by himself, "within five years" [65]. It was, in fact, not until the early '70s that the fields of Computer Science and Artificial Intelligence realised that problems in vision are, in fact, difficult[89].

2.2.2 The Reality

To take the example of Chess, by 1968 chess programs were doing well, one notable chess program by Richard Greenblatt being given an official ranking of 1640 and official membership to the U.S.C.F. and Massachusetts Chess Associations. By 1985, home computers had chess programs which rankings as high as 2200, i.e. Grand Master Level. In 1997, IBM's Deep Blue defeated the world chess champion, Gary Kasparov, proving that in certain tasks, computers could perform as well as humans, despite their different approaches. To take other examples, speech production was available to the consumer as early as 1986 with speech software for the BBC Microcomputer, and dedicated speech hardware for the ZX-based Spectrum computers. At the moment, there is Optical Character Recognition (OCR) software freely available for scanners, and text processing software built into Word Processors to catch grammatical errors.

However, if we consider the agility of human motion, the range and robustness of human hearing, and the adaptability and power of human sight, we can see that no system has come close to challenging human achievements in any of these physical fields in the same way that the games-playing machines have in the mental one[90].

As can be seen, history unfolded contrary to how the early experts predicted. It seems that very cognitive tasks, i.e. tasks which require pure processing power, are more easily overcome than tasks which require reasoning, and/or motor control.

Therefore, we are faced with a paradox. If we are so adept at creating cognitive simulacra, why have tasks such as vision, a seemingly purely cognitive act, proven so hard? Ullman states that the proficiency of the human system in analysing spatial information far surpasses the capacities of current artificial systems[90].

2.2.3 Why were the predictions wrong?

One might wonder why such errors were made. Some, such as Brown believe that the problem was due to the non-introspectable 'black-box' nature of the brain [84, 89]. Others, such as Ballard and Brown hypothesise that the volume of data which required processing was the stumbling block [34].

Some believed that they simply had to wait for computers to get fast enough, and have enough memory. Poggio points out that there is a duality between computation and memory. Given infinite resources, the two points of view are equivalent[54]. For instance, one could play chess by precomputing winning moves for every state of the chess board. Assuming infinite resources, or a 'sufficient' amount of time to precompute the result is unreasonable, however. *Ad hoc* methods and tricks, such as pre-computation have consistently failed[89].

Claude Shannon founded Information Theory in his 1949[59,91] paper. He calculated the number of board states in chess at roughly 10^{120}, a number higher than the number of atoms estimated to be in the universe, proving long before the problem of vision was first tackled, that not all AI problems were solvable by pre-computation and an assumption of infinite resources, therefore another way of solving such huge problems was needed.

The early chess-playing programs, in their reliance on brute- force and machine speed, borrowed little from what was known of human chess playing processes. The clear demonstration by their weak levels of performance that speed was not enough, produced a gradual movement towards incorporating some of the task-dependent heuristics that humans rely heavily upon[80].

In the 1950s and 1960s even the largest computers had performance inferior to a modern low-cost desktop PC, and the implementation of software capable of playing the game of chess at even the most naive level seemed daunting.

Chess's complexity comes from the combinatorial explosion that happens after the first move. With that combinatorial explosion, the computation required to perform decision making increases proportionally. Humans have always been pessimistic about computer advances. It is clear from programs such as "Star Trek" in which computers were simple tools, that even by 1969, people did not expect the field of computing to progress very far, even by the twenty-second century, and hence the estimated computing power needed to play chess, as well as other cerebral tasks must have seemed an impossible obstacle.

In both fields, it was recognised that the quantity of data which required processing was too large to cope with. In chess, the game was divided into *plies* or turns and the notion of *minimaxing* whereby the program attempts to minimise the opponent's advantages, while maximising its own, were used. In stereo vision, the images were reduced to tokenised descriptions, and the scenes were limited to Blocks' World or structured environments.

Marr realized that a science of visual information processing was needed, and much can be learned from the breadth of his approaches and his rigorous detail in the analysis of specific problems[89].

At first, the chess and vision problems seem to be similar. Both have some input data, and require some algorithm to be run to extract meaning from it, and produce some possible actions. However, the main difference becomes evident when one considers the amount of information at each decision step.

Each board state in a chess game can be compressed into 64 bytes (an 8x8 board with one of 7 pieces, or a space). Each image in a stereo system is typically 256x256 pixels at 8 bits per pixel. Since binocular stereo uses two such images, there is therefore 128 kilobytes per scene, roughly two thousand times as much information to consider per problem! Erken states that most of the computational difficulty [in early visual processing] lies in managing the incoming flood of sensory data[76].

Immediately realising this, researchers implemented only the simplest of algorithms in the first feature-based matchers, usually directed towards reducing the representation from many kilobytes down to a few hundred bytes. At first, they were rewarded with remarkably good results. The traditional viewpoint when designing 'early' or low-level vision solutions is that a fast approximate answer is more valuable than a slow correct one[92]. With respect to this, Beiderman discusses the simplifications made in the early days of vision in order to achieve this, and shows that the original simplification to a Blocks' World scenario was too unrealistic, raising hopes falsely and stagnating research [93]. Choosing appropriate features to tokenise is difficult[75], and token matching stereo algorithms have a hard time reconstructing surfaces because of the sparcity of the tokens[29].

By 1973, it had become clear to some that if the skilled man is an intelligent man, he is also a learned man[80], and Semantic Reasoning and Knowledge Based Systems offered a way of adding such world-based knowledge to aid the stereo matcher. Vision researchers began to marry the two techniques with computational vision, and hence the model-matching and functional vision techniques came about [94]. In his review of the field in 1988, Brady reported that model-based systems were 'narrow minded', were ineffective when there was too much extraneous clutter, and lack any 'graceful degradation'[32]. While these systems had moderate success, most were extremely limited to a small subset of objects they had information about, and because of the inability to give them 'common sense'.

So it can be seen that by classing vision as a purely cognitive process, and using the same data reduction techniques which had produced such advances in chess and other AI fields led researchers into believing that they were making great progress.

It was only when it became apparent that the reduction of complexity applied to vision to bring the combinatorial problem in line with the chess solutions created systems so weak that they were too fragile for use outside the laboratory that the true problem of the SCP became apparent.

2.2.4 The Contributions of Marr and Poggio

In 1968, when a computational approach to chess was effectively solved by Richard Greenblatt, stereo vision was not advancing as quickly. At the time, researchers such as Minsky were still casting about for ideas, as demonstrated by his internal memos at MIT [77,88].

Work continued steadily, and by the mid-1970s it was being tackled by three schools. *Neurophysiologists* attempted to understand how sensory and neural mechanisms of biological systems function. *Perceptual Psychologists* tried to understand the psychological issues governing the task of perception, and *Computer Vision Scientists* investigated the computational and algorithmic issues associated with vision[95]

Perceptual psychology and neurophysiology were revealing much information about the functioning of the eye and brain. The belief was that studying the computations that underly the brain's competence may lead to the development of new, more efficient algorithms[90]. Leaders in the field of AI, such as Marr, Poggio and Grimson, disgruntled with their own crude results, began to re-evaluate their views on stereopsis, based on this new data. In 1976, Marr and Poggio produced a paper "Cooperative Computation of Stereo Disparity"[36], which used a connected 'network', in much the same way as used by this project. The concept was elegant, simple, and robust to the "loading rules" given to the inter-nodal dependencies. The amount of computation required was large, but optimisable. By October of 1977, they released a further paper with Palm, analysing their algorithm, and showing some extremely promising results, and affirming the claims of its robustness, prompting an avenue for research[61].

Within a month, however, Marr and Poggio had released another set of memos and papers concerning their new Spatio-Frequential theories. One was finally published in 1979, "A Theory of Human Stereo Vision" [1], also known as the "second algorithm"[96, 97], another landmark paper which began the work into the tackling of the SCP using knowledge gained through frequency-based methods.[96]

Due to the inability of the field to produce a satisfactory solution, this paper proposed a new biologically-inspired algorithm. It separated the stereo vision process into five stages,

1. Images are filtered with four sizes of masks, given by $\nabla^2 G$.
2. Zero Crossings are found along Horizontal Scan lines
3. Similarly oriented Zero Crossings are matched at the various filter levels.
4. Coarse levels of matching are used to aid Finer levels.
5. Matches are stored in a dynamic buffer known as the $2\frac{1}{2}$d sketch.

The concept of matching based upon frequential "channels"[1, 98], was novel, and paved the way for the Spacio-Frequential methods. Grimson went on to implement their system in 1980 proposing some modifications. Since that time, many papers have implemented similar systems, and some are discussed later[22, 28, 29]. Grimson, however acknowledged that there were flaws in the algorithm but never implemented his suggested remedy [96, 97].

Because their theories were based on research done by neurophysiologists, those working on these systems claim psychological validity[1]. In fact some neurophysiological studies carried out since have suggested close links between stereopsis and phase information[29, 80, 99], which has led to many stereo algorithms which operate fully in phase-space. The only drawback to this such an algorithm is difficult to implement efficiently on a general digital machine[29].

Most of stereo vision is still divided into the correspondence-based feature, the correlation-based area, and the Spatio-frequential camps, with most current work being carried out in the Feature-based areas[4]. Since those early days, the consistent inability of the theories to explain and emulate efficiently certain aspects of biological vision, new techniques have emerged, such as shape-from-shading, which fall into a new fourth category of *Shape-from-X*[53,67,72,100].

Depending upon the 'X', the algorithm may be feature, area, frequential or a hybrid. This makes the shape-from-X systems difficult to classify. Compared with stereo vision, shape-from-shading or motion relies on complex sets of prior assumptions[15], and hence most are viewed as higher-level modules to be added to a pre-existing 'assumptionless' stereo system to add extra functionality, rather than be stereo systems in their own right.

Such hybrid systems are gaining acceptance. Fua[101] discusses the fact that stereo is robust in textured regions, but potentially unreliable elsewhere, whereas shape-from-shading is reliable only in areas where there is little texture. Integrating such modules together is a formidable problem[72], and not as simple as Marr's proposed integrable modules would hypothesise[65]. It has been long realised that there is no "right" or "wrong" way of computing stereo, a given computation can, in general, be performed by several different algorithms[92]. Therefore, despite the neurophysiological evidence for Neural Networks and Spatio-Frequential Methods, work has still continued in the classical stereo methods.

2.3 Capturing Stereo images

Although the standard camera arrangement for *human* stereo vision involves two eyes side-by-side, many other systems exist in the world. Spiders, for example, employ typically 8 varying-range sensors to monitor their surroundings. For this reason, research has not been solely confined to studying *Binocular Stereo.* Much work has been done in *Multiocular stereo*, employing 3, 4 or perhaps more cameras. Motion stereo (typically) involves a single camera, moving around an object to capture views from all sides[67], or by a camera translating through a scene, or a static camera with moving objects [102]. Since motion stereo uses more than two stationary image frames, it usually gives more accurate depth measurements than static stereo which uses only two[102].

During the 1950's a lot of work on stereoscopic scene modelling was carried out for the glasses-based cinema systems of the period [103, 104]. This work laid the foundations of that done in the 1960's, where the focus changed to the single-user and automated stereo, thanks to the CCD camera technology becoming available, which was crude, but allowed scenes to be processed immediately without the need for film processing.

Camera technology advanced greatly during the 70s and 80's[32], and there are now high quality CCD and CMOS cameras with which to take far clearer and higher resolution colour images. The commercial machine vision industry emerged in the early 1980s. Commercially available applications to date have been mostly in automation of inspection and classification of objects in controlled settings[76]. Using humans for inspection requires considerable manpower that significantly increases the costs and the time for inspection[62].

Brown reports that bi- and trinocular robot heads are increasingly available with three to twelve controllable degrees of freedom including intrinsic (focus and zoom) as well as extrinsic (pan, tilt) parameters[84]. Brown goes on to say that low-cost modular hardware can nowadays provide frame-rate (60Hz, 512x512) colour digitisation, pixel level processing, convolution, feature-detection and many other basic capabilities.

Despite the early problems with the camera technology, researchers did much work identifying the best operators for spatial feature analysis. The work by Canny[105], and Marr and Hildreth[106] in identifying good sets of edge-detection operators is still widely used, and almost twenty years after their introduction to the field, few edge operators perform as well as theirs, despite their poor response to noise.

Recently, colour CMOS and CCD cameras have become available which are of a standard equal to their grey-level counterparts. Colour not only adds to the sensory richness of perception, but is also an important information carrier, and is an important property for object discrimination. Colour is being increasingly used in machine vision in recent years[107]. Although colour is not vital to stereopsis (proved by the earlier successes with grey-level cameras), it does occasionally fill out the description of a scene making stereopsis easier[39].

Depending upon the problem to be solved, the choice of camera arrangement is crucial. Some of the typical camera arrangements and their advantages and disadvantages will now be discussed. In the next chapter, the algorithms which must work on the data provided by these systems are discussed.

2.3.1 Monocular Stereo

Monocular Stereo may seem like a contradiction in terms, perhaps the term Monocular Depth Recovery is more accurate. Some systems using model-based matching or which take a series of frames for motion stereo can achieve stereoscopic results[108]. From a study of the literature it is clear that model-based matching is, at present, of very limited interest, since systems are restricted to a small number of pre-trained objects. Dorai and Jain developed a novel system, COSMOS which described objects in terms of curves, and attempted to classify the objects under scrutiny according to the curves it 'saw'[109]. This system was excellent; however as the number of objects in its database increased, the certainty of the classification decreased dramatically. Also, monocular vision for the recovery of 3-D information is error prone under variant lighting conditions[44].

When complex objects are involved, or when classification is uncertain, active camera motion may be required also[110]. This *Motion Stereo*, because of the varying numbers of cameras that can be used to obtain it, is described in a section by itself later.

2.3.2 Binocular Stereo

When one considers *robotic* vision, the image that comes to mind is usually of two cameras situated side-by-side, in a similar manner to the human visual system. Figure 2.1 shows a typical layout for a mobile robot equipped with a stereo vision system. The input from the cameras is fed to an on-board computer which processes the information and uses it to drive the motors.

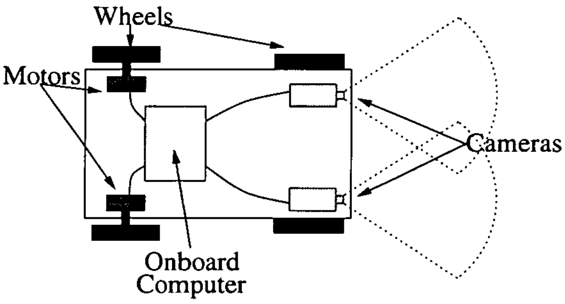


Figure 2.1: *A Typical layout for a stereo-equipped mobile robot*

This is the traditional layout of stereo vision, and it is the aim of this project to use such a system, although not mounted to a robotic rig. There is biological evidence to support the hypothesis that binocular stereo works in every two-eyed creature that can successfully navigate its environment, which includes humans. The cameras do not need to be side-by-side, of course. If the cameras are coplanar, i.e. conform to the horizontal epipolar line constraint (which is not an easy task in real-life situations [74], this means that Pixel Y coordinates have no effect on the estimation of depth [18], and therefore vertical information is not quite as easily recovered as horizontal information. This leads some to say that traditional systems provide only $2\frac{1}{2} - D$ results[65].

2.3.3 Multiocular Stereo

The *tri*nocular approach to the stereo problem has been proposed recently as an alternative means to conduct the correspondence search[4]. In fact Kanade showed that by increasing the number of cameras, the problem can be simplified[71]. In an earlier paper, Dhond and Aggarwal found that an increase in the number of cameras, which led to an increase in the computations required, also led to a dramatic reduction in the number of false matches[111].

Multiocular stereo is also useful in situations where the *depth field* (i.e. the maximum and minimum distances at which an object must be stereoscopically resolvable), is extremely variable. In this case, one can have a multi- baseline stereo system, as shown in Figure 2.2, which can resolve stereo at short, medium and long distances, as well as extract some vertical information as well.

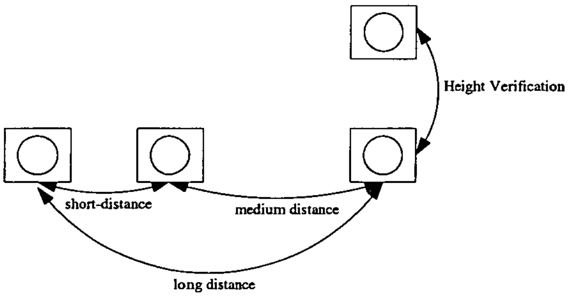


Figure 2.2: *A Multiple Baseline Stereo Rig*

Multi-ocular stereo is popular because it provides more information than a binocular system, thereby reducing false matches, and increasing likelihood of obtaining a positive match while still maintaining passivity. It is not only in battlefield scenarios where active emissions, such as ladar[1] and radar are undesirable. In scenes which contain shiny metallic surfaces, a system employing a ladar, for example, would have problems with specular reflectance [31]. Laser-based sensors are also difficult to use when the target objects have a variety of colours, or are transparent[62].

[1] A term to denote a laser used for range-finding

The work by Okutomi *et al* over the last ten years has generated some excellent multi- ocular systems using the Sum of Sum of Squared Distances (SSSD) approach [3, 56, 112].

2.3.4 Motion Stereo

Motion stereo infers depth information from a sequence of image frames[102]. It is sometimes known as depth from stereo image flow[113]. Perceptual studies suggest that the human visual system relies on some sort of motion analysis to perform the task of understanding the environment[114]. The most well-understood meaning is that of using optical-flow, say in motorway situations, to work out the distance to objects and their relative speeds. However, the term also applies to a single camera (or multiple cameras) traversing a scene and using the disparity between frames (much like standard binocular or multiocular stereo) to calculate depth. Differential techniques and region-based matching are well known methods for computing optical flow[115].

Both Zhou and Kato *et al* use a single translating camera to locate edges in a scene more accurately. However, Kato uses an edge-detector, and Zhou uses a Neural Network [102, 108].

Motion stereo in itself is very useful, since it allows us to understand how humans can use Spatio-temporal representations to keep track of moving objects etc. At some later point, it may be useful to use a motion tracking module as a back-end to the system developed in this project in order to improve inter-frame matching, and to assist with the SCP.

2.3.5 Mixed-mode stereo

While this project does not deal with active stereo, it must be mentioned that using only cameras is considered a "single-cue" approach.

Single cue approaches often produce unreliable interpretations, and fail due to insufficient disambiguation power[27]. The existence of false targets is one of the most serious problems for matching[1, 96]. One possible option for failure detection in stereo matching is to adopt another sensor exclusively utilised for the purpose.[81]. Nishigaki states that it is indispensable to establish a reliable method for the detection of failure[81].

Molton, for example, in his stereo rig for assisting the visually impaired reports that he chose the combination of sonar and vision because of the complementary nature of the two sensory modalities[116]. Sonar measures range accurately, but not direction, whereas stereo vision measures direction more accurately than range. He also uses an inclinometer and a compass to measure direction and tilt changes as the wearer of the system walks. Mixed-mode stereo is not common, and only several papers have been found covering it.

2.4 Stereo Algorithms

Many of the original vision computers did many tasks using specially designed hardware processing boards. With the accessibility and industrial open-architecture aspects of the VME bus, there was rapid prototyping of hardware signal processors[117].

Unfortunately, the camera hardware of the day was not as advanced (or as modifiable) as the computing hardware. Cameras were not a medium which could be manufactured in the laboratory, as easily as DSP boards, and the stock cameras available added substantial (sometimes unacceptable) amounts of noise due to the youth of the CCD technology.

Due to the inadequacies of camera technology, the researchers working on spatial algorithms encountered a paradox. Matching on a per-pixel basis can generate many hundreds of possible answers which have to be compared in order sort out the right answer from the wrongs. Correspondence search is time-consuming, even if using state-of the art general-purpose processors[49, 50, 118]

One can therefore attempt to match blocks of pixels in the hope that a group of pixels is less likely to have as many exact matches. The smallest block is a single pixel, while the largest possible is the size of the input image.

There are many advantages and disadvantages of block-matching; however choosing the right size of the block is crucial. The likelihood of obtaining a unique solution is proportional to the size of the block. However, as the block size increases, some problems begin to occur.

- A block does not always match in whole. The larger a block, the higher probability of the block incorporating pixels which are wrong, but merely in the minority. When this occurs at an object boundary, it is known as "boundary overreach"[56]
- The computational load increases proportionally with block size.
- The effects of parallax and foreshortening can cause blocks of pixels to have no recog-

nisably matchable equivalent.

Small windows are more appropriate for a disparity change region, and large windows are more appropriate for disparity-smooth regions[5, 7]. The simplest way around this problem is to use variable-sized windows. The adaptive windowing techniques of Okutomi and Hariyama are good examples of this[50, 56]. Fixed-size-window techniques are highly parallel in nature, and therefore highly optimisable. By using variable window sizes, however, pixel-level parallelism is not suitable due to low utilisation of hardware[50].

As can be seen, even with a task as simple as choosing the size for a block for matching, there are significant issues which must be taken into account. The whole domain of stereo vision seems to be based upon trade-offs. The following sections will list a few of the typical choices faced by a vision researcher, and the constraints which guide the choices.

2.4.1 Modular or Integrated Stereo Vision Algorithms?

Marr discussed the idea of separate 'plug-in' vision modules[65], however Pankanti discusses this and dismisses it as a "gross simplification of the reality", and opines that the general integration of these modules is a formidable problem[72]. Whether these modules will ever be as swappable as Marr envisages is subject of constant debate, however most of the techniques discussed in the traditional AI and vision literature have a standard input and a standard output, and hence are far easier to integrate than other disciplines. Some researchers, such as Gamble integrate the modules in a weighted-sum sense, to gather a 'consensus result' from the modules[12, 119], rather than some of the more exotic methods discussed by Pankanti, such as Lattice-Theoretic and Game-Theoretic frameworks[72].

Therefore, although many of the techniques described in this and the following chapters have been divorced from each other for clarity, it should be kept in mind, that most systems employ the services of differing, and often opposing, modules, held together using various integration models suited best to the particular interaction[72].

2.4.2 Spatial or Frequential?

In his recent paper, Cox compares the predicted path of Vision, and the current bias of research and shows that while frequential methods are producing some good systems, by far the majority of researchers are investigating the spatial solution. This led him to speculate that: “Spatial methods look like the most likely to result in a working generalised vision system”[64].

Why is this so? Investigating the way that technology has advanced, compared to its state in the sixties, it can be seen that computers have become far less accessible to the researcher on a hardware level, but have become much more powerful on a software level. General computer theory has not changed significantly, however, and experimentation during this project has shown that unless large amounts of time are spent highly optimising the code, it is still faster to prototype a system for, and perform raster (spatial) operations on, images, than it is to convert the data into Frequency space, and convolve there. Jepson reports that while their Phase Differencing system was easily implemented on an analog machine, it was very inefficient to implement on a digital serial computer[29].

Software solutions are more rapid to prototype, and easier to customise to differing roles than hardware ones. With newer high-quality cameras, and with the advent of new techniques which are insensitive to, or counteract the photometric variances, the original reasons given to support frequential methods over purely spatial ones are reduced.

2.4.3 Brute Force or Heuristic? - Can they be separated?

If we examine a problem closely, there are always two ways of solving a problem such as this. The first involves the so-called *brute force* searching, in which the computer rapidly searches through all of the possible alternatives until the best one is found. The second is known as *heuristic* searching, where some notion of what would be a good move, and what would be a bad move is given to the machine, and it scans quickly through the ones it considers the best. This is a corollary to the computation/memory duality given above.

When a computer is confronted with two images, according to the Stereo Correspondence Problem, it must search through the two images for matches, and hence work out the depth, with no *a priori* information about the scene and no operator assistance[120]. In other domains, such as game-playing, the validity of each hypothesised move can be tested directly because each board state can be reduced to a table of threat and advancement values.

Similarly, a Brute Force Vision system must test each solution it works with, however the method of traversal through the solution space, coupled with the verification method can greatly affect the speed of the system[51]. With respect to the verification method, Grimson points out that there is no constraint that the sensory data for one problem must come from one sensory modality[121].

Whichever way is chosen, the brute force system must test each and every solution possible in order to prevent local minima. However, it may not be clear in all cases which solution is better than another, and the only way to be certain is to actively scan the scene, using a laser, radar or some other system, thereby removing the benefit of a passivity. This 2-D full search is a very common method in early vision systems[122] because of its regularity of memory accesses etc, however is very computationally expensive[7, 50], and non-optimal. As can be seen, the disadvantages of a passive system are that it requires many photometric assumptions and has a high computational cost[5].

We must therefore use some heuristic elements. Some searching of the solution space is necessary for any system; however most systems use large amounts of knowledge-guided data, constraints and other heuristic processes to reduce the space that must be searched.

2.4.4 Algorithmic Constraints

In the task of Stereo Matching there are many trade-offs, not the least of which, is deciding how general to make the final solution. If the aim of the project is to solve the task of general purpose vision with no constraints, then it becomes daunting, and at present impossible. There are scenes that even the human visual system with the power of the brain cannot decipher, i.e. *optical illusions*. Therefore one has to constrain the demands placed on the final system. Although each project investigated in Chapter Three uses many constraints, there are some which form the core agreed-upon constraints of the field.

Blake and Zisserman also introduced the notion of Weak Continuity Constraints, whereby one could choose to ignore a constraint if the evidence to the contrary was strong enough[123, 124]. A similar ideology based upon weighted supporting evidence is used in the project.

Uniqueness Constraint

If one considers the trigonometry of a typical stereoscopic rig, as shown in Figure 2.3, it can be seen that the image of the point P, as projected onto Pl in the left image and Pr in the right image must pass through the focal point of the camera. Since there is only one point P, and one focal point per camera, there can be only one image of P in each camera image[9]. Therefore for each pixel in the left image there can be a maximum of one matching pixel in the right.

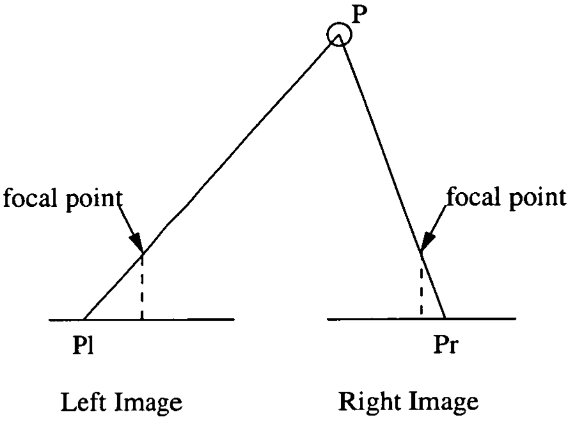


Figure 2.3: *Uniqueness Constraint*

This, of course, breaks down if mirrored or transparent surfaces are present. Here, relying upon the Uniqueness Constraint means that one restricts oneself to only dull *Lambertian* surfaces. This greatly reduces the amount of processing required, and for most of the jobs that the system might be used for, is perfectly valid. Work has been done on reflective and curved surfaces, such as the work by Bhat and Nayar[20], where use is made of specular reflections. Current stereo algorithms are seriously deficient in dealing with specular reflection[20, 72].

Smoothness Constraint

In the real world, objects can take any shape. Most systems, when confined to the laboratory, are tested on regular shapes, such as cubes and spheres, the Blocks' World simplification. For a useful product, the system must strike some balance between these extremes. Consider the images in Figure 2.4. The cameras are shown at the left, and are viewing the object on the right. The rig is seen from above. Where possible, viewing zones in which a feature of the object is visible in one camera but not the other, are shown as a light grey field.

These objects are perfectly viable, but unlikely to occur in a real life situation. However, attempting to build a system that would know how to deal with these objects is not practical. If a human were faced with such a problem, they would stand back and take a vantage point which had more correspondences. On a mobile robot, or a system which uses active camera motion, this may be possible, but a purely passive system cannot move away from, or rotate the

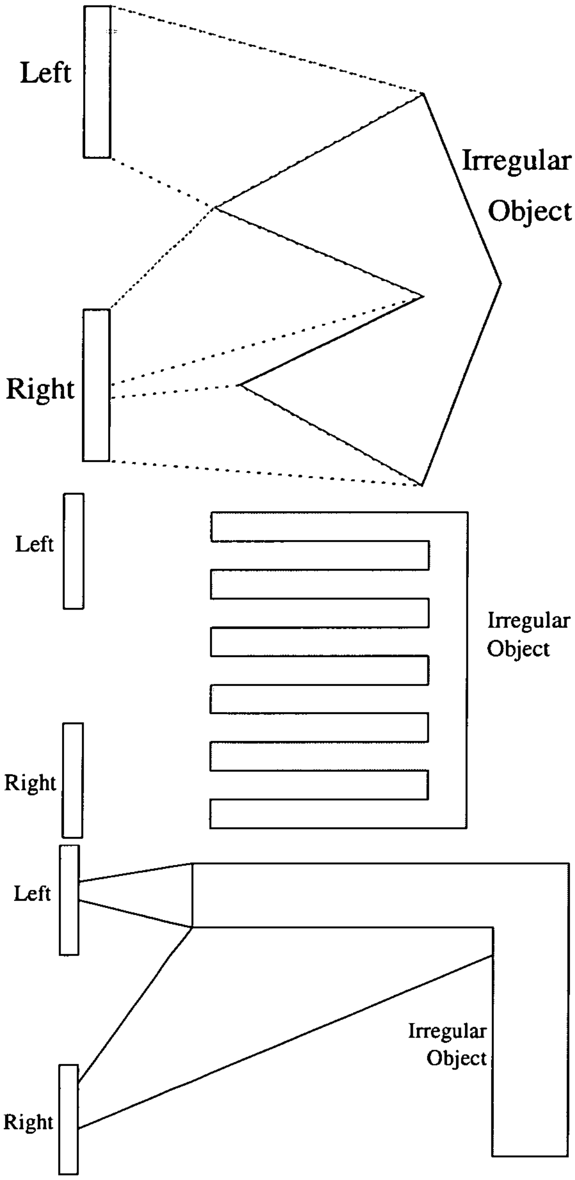


Figure 2.4: *Objects that violate the smoothness constraint*

object for a better view. Therefore, until the system is able to control its own viewing angle, the smoothness constraint is enforced which, as the name suggests, says that an object in the scene must be smooth, i.e. it must not have large changes in depth, and must not have 'holes' in it.

Coherence Constraint

The coherence constraint is a slight adaptation of the smoothness constraint. It says that if an object is matched, and there are two solutions, one of which is a solid block, and the other which contains 'holes', one should choose the solution where the match is solid. Figure 2.5 is divided into three possible matches. Matches 1 and 2 are the usual results from a naive system and match 3 would be the result from a system that allowed the matcher to cleave sections of segments if a good solution under the constraint did not present itself. Under the Coherence Constraint, Match 1 would be chosen over Match 2, which may in fact be the wrong choice. However, this constraint's main use is in identifying such problems and enabling systems to divide up segments to better fit the constraint, and therefore a less naive system would choose Match 3.

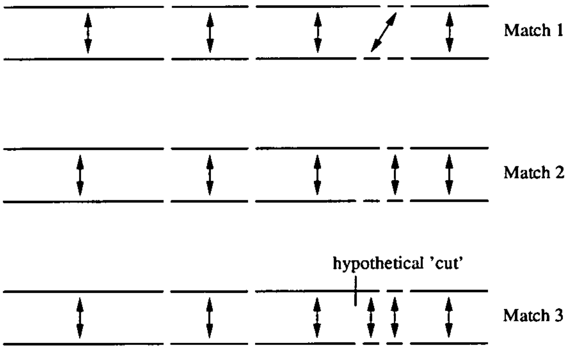


Figure 2.5: *Matching line segments under the coherence constraint*

Epipolar Constraint

When considering the SCP, one can see that given a little knowledge of the scene, and the geometry of the cameras, the area that one has to search in order to perform a match can be reduced. In fact, the dimensionality of the search space can be reduced from two dimensions to one dimension by observing that for any point in one image its potential matching points in the other image must lie along a line[16,24–26,41,51,67,74].

This is the epipolar constraint, a geometric property available with the stereo sensor whenever it is properly calibrated [14,43,45,58]. Figure 2.6 shows two converging cameras and an epipolar line through them.

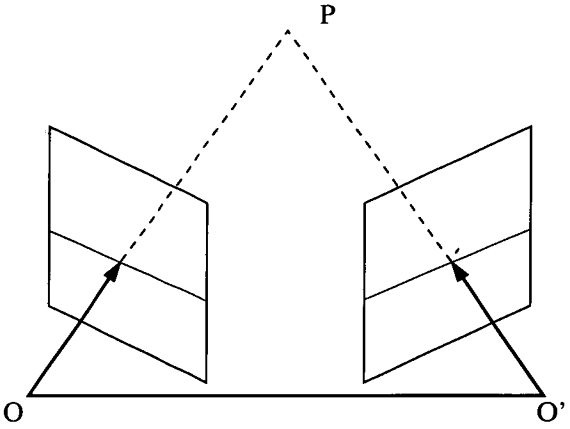


Figure 2.6: *Epipolar Constraint*

More Formally, the epipolars are the intersections of the image planes with the plane defined by the two camera origins O and O' and the feature point P that we are observing.[17] Hence the projection of a feature point P that we are observing in one image must be on the epipolar of the corresponding feature point on the other image plane.

If the cameras are coplanar, with their recording surfaces' y axes aligned collinearly with the World Y axis (upward), as shown in Figure 2.7 then the epipolar lines are equivalent to the raster scan lines of the images themselves, which accelerates computation considerably[73].

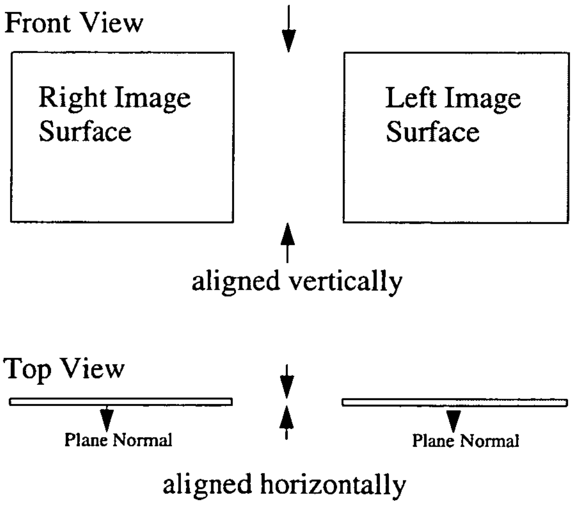


Figure 2.7: *Coplanar, Collinear Image Surfaces to Simplify the Epipolar Constraint*

Epipolar geometry plays an important role in stereo vision. It can be computed between two uncalibrated images by a certain number of corresponding points[14]. When the camera calibration is calculated, and the relationship between the cameras expressed mathematically, this leads to the *essential matrix*[48], a matrix which defines the rotation and translation between the cameras

If this constraint it met, then we can go on to make further simplifications, such as the fact that an object in the left eye (or camera), which casts an image at point x_l and an image at point x_r will do so such that $x_l > x_r$. Put simply, an object always appears further to the right in the left hand image. This removes on average 50% of the epipolar search space, and leads to the Ordering Constraint[9, 25]. Dhond and Aggarwal give a complete description of the epipolar constraint, and its implications[4].

Chai points out that the epipolar geometry constraint is the only geometrical constraint available regardless of specific objects[14]. By this, it is meant that the layout of the cameras can be controlled by the researcher, however the objects cannot.

2.4.5 The final choice

As has been shown, there are many different choices to be made when solving the SCP. Some are quite complementary, and others are difficult to integrate. There are, however, always constraints which can assist (or hinder) in the choices, to render the problem less "ill-posed". Much research has been done into other systems created to date, and almost all combinations of the above methods have been designed, and almost all adhere to the constraints given. Chapter 3 will discuss some of the major solution groups.

An important consideration is how the data is to be presented to the user. For example, there is little point recovering a dense depth map if only an audio cue, for example, will be used to denote when an object has approached too close to the sensors. In this case, a sparse depth map is sufficient. At an early stage in the research, work was undertaken to investigate the availability of appropriate display technology. This work is summarised in Appendix B.

2.5 Summary

This chapter explained the history and present-day status of Artificial Intelligence, Computational Vision and Stereoscopic Display research. As has been shown, the fields have taken many changes in direction, thanks to the input of biology and computer science, as well as the mass of research done within the field, and the economies of implementation necessitated by the hardware available.

Because of the enormity of the task, and in order for the solutions to be fast and practical, but not considered too limited, there have been a number of agreed-upon, common-sense constraints imposed, some of which have led to a stagnation in research, whereas some have benefited AI vision immensely.

The conclusion of this chapter should therefore be the axiom: "There is more than one way to do it"[125]. and that adequate evidence occurs for all of them. However the conventional approach seems the current favourite amongst researchers, even though it may not seem as elegant as the newer ones.

Little has been mentioned of the low-level algorithms themselves. Therefore, Chapter Three will discuss in detail many of the common "modules" or algorithms in use, their advantages, and their disadvantages.

Chapter 3
Stereo Vision Techniques

3.1 Introduction

In the previous chapter, the concept of 'pluggable modules'[65] (or "Visual Routines"[90]) was discussed. Poggio uses the hypothesis that the brain uses such modules in his 1990 paper on the functioning of the brain [54].

While not as 'plug and play' as Marr envisaged, such modules do exist, and are in use. In this chapter, a selection of the most used modules will be described, with the aim of showing why they were or were not included into the project.

After the modules have been described, each of the three main approaches, i.e. Spatial, Frequential and Connectionist will be taken in turn, and a representative sample of papers from that domain described, compared and contrasted.

3.2 Computational Modules and Algorithms

Stereo suffers from many problems: discontinuities within windows, occlusion, noise, specular reflection, projective distortion and camera variations [4, 58, 68]. Each stage in the correspondence part of the stereo pipeline attempts to tackle one or more of these problems.

Noise presents one of the biggest problems to the vision researcher. It is impossible to avoid its undesirable presence[114]. Noise occurs for many reasons; RF interference, camera errors, digitisation errors and so on. The consequence of noise within a vision system is that the system becomes inaccurate due to difficulties attempting to match noisy pixels, and can therefore exacerbate other problems. Many modules are designed, or have evolved, to be noise-tolerant, but those which are intolerant require the images to be passed through a noise-reducing process beforehand. Smoothing and filtering are methods by which noise can be reduced, and will be discussed first.

After cleaning, the next stage in the pipeline is typically that of detecting elements within the image in order to perform the matching. Block matching and edge-detection are the two most common methods employed. Corner descriptors will also be discussed since they are popular and more robust.

In more complex systems, model matching or shape recognition are used also to assist in the stereo process, and while too complex for this project, were researched and fall within the bailiwick of stereo vision techniques, and are therefore discussed.

3.2.1 Smoothing and Filtering

In noisy environments, smoothing can help to reduce the number of irregularities within the image. Interference noise can be effectively cancelled by a simple small gaussian smoothing (see Figure 3.1b), or a neighbourhood-averaging kernel (see Figure 3.1a).. Impulsive noise, as caused by a faulty CCD surface for example, can be removed using a median filter. Depending upon the sizes of filters being used, different levels of smoothing and filtering can be achieved, allowing the process to be well controlled and well understood.

Noise can be characterised as unwanted high-frequency information overlaid onto the images. Smoothing solves this because it removes **all** high-frequency information. Unfortunately, much of the informational content of the images is contained in the higher frequencies, and is therefore removed too[65]. If, after smoothing, one is performing feature matching using edges as pivots, one must beware that edges are high-frequency elements, and are removed or diffused by a smoothing algorithm. The size of filter used is very important. Unfortunately, in the case of impulsive noise, the smoothing filter has the effect of spreading the anomaly's effects over an area, corrupting previously correct pixels.

Median Filter kernels construct a list of the pixel intensities in the immediate neighbourhood of each pixel in the image. The value of each pixel is replaced with the median value of the intensities of the pixels within its neighbourhood. The effect is similar to the traditional smoothing filters, however also removes any impulsive noise from the input. This is obviously a very desirable property to have. However the median filter has a longer runtime due to the list sorting necessary. Figure 3.2 shows a median filter removing an element of impulsive noise from a 3x3 block of data.

Smoothing does not only remove noise. It can also assist in the reduction of differences between

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

a)

$$\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$$

b)

Figure 3.1: *Two 3x3 smoothing kernels: a) Standard Neighbourhood averaging Kernel b) Gaussian Kernel*

12	19	9
18	89	7
22	19	13

a)

12	19	9
18	18	7
22	19	13

b)

Figure 3.2: *The effect of applying a median filter to a 3x3 neighbourhood: a) original b) median filtered*

the stereo images due to quantisation errors.

Some level of smoothing is essential in any vision system since perfect, noise- free cameras are not available at the moment. In any case, it is unwise to rely on perfect data anyway, in a world where simple things such as mud on the lens could easily falsify this assumption.

3.2.2 Block Matching

Attempting to achieve a correspondence between stereo images at the individual pixel level does work to a degree; however featureless areas, repetitive patterns, foreshortening, occlusion, photometric variances and noise all pose significant problems. The reason for this is that there is insufficient information at the pixel level for reliable verification. Boyer suggests that pixel-level matching is useful, but only as a fall-back algorithm to fill the inevitable featureless gaps left by multi-pixel or feature-based algorithms [59].

One of the simplest solutions to this, is to group pixels together into blocks, and attempt to match these blocks. Block matching is a widely used method for stereo vision, visual tracking, and video compression[126]. The 2-D full (or exhaustive) search method (2DFS) dominates most parallel hardware implementations because of a high degree of regularity in its memory accesses and computations. As Chen points out, however, the number of computations necessary is prohibitive. For a standard video-conferencing application, typically over 800 million subtract/accumulate operations and over 1500 million data accesses per second are required.

Various fast search algorithms, based upon the assumption that the mean-absolute-error (MAE) between the current frame block and the previous frame search candidate increases monotonically as the search position moves away from the best match position, have been used in the past to alleviate the computational burden [122]. Using an MAE or mean-square-error (MSE) metric, the monotonic assumptions and the epipolar constraint to limit the number of possible matches, provides a robust, relatively rapid and noise intolerant method for stereo matching. However, most of the fast block-matching algorithms do not guarantee that the match found is the globally optimal match in the search range[126].

Unfortunately, there is a drawback to the block-matching algorithm which no assumptions can alleviate. When designing the block-matcher, one must choose the size of the blocks with care[49, 56]. Usually each image is divided into non-overlapping blocks. The motion vectors [disparity] for all the pixels in one block are treated as one single vector to be estimated[126].

A small block-size results in a faster algorithm which produces more textured depth maps, and is much less likely to be fooled by step-edges. However small blocks are much more susceptible to false and multiple-matches. Larger blocks are much more discriminating, and produce fewer candidate matches, and subsequently, fewer false-matches. Unfortunately due to the sizes of the blocks, small features, such as edges tend to become smoothed over[5, 7], and the computational burden of calculating so many squares and averages of large blocks produces a much slower solution. The main problem with larger blocks, however, occurs at step-edges, where the block can either completely smooth over the edge, distorting the object, or can get confused and refuse to match the block at all due to parallax differences and occlusion.

Several researchers have suggested adaptive block-sizes to overcome this problem, and have had good results[7, 49, 50, 56].

3.2.3 Edge Detection

The amount of data present in each image which requires matching poses a significant problem. The simple fact that photometric invariances and projective distortion can alter the appearance of an object between the images further compounds the problem.

In an effort to overcome this, unusual or easily-detected pixel formations, or *features*, are marked with tokens in each image, and these tokens are subsequently matched. The matching at this level aids the matching of the rest of the data. Extracting significant features from images is one of the main difficulties of stereo[127].

Since feature matching forms the pivot of many systems, it is important that these features are **robust**. The process by which image pixels are classified as feature or non-feature points must be *rapid, repeatable* and *selective*. Repeatability and Selectivity are the two most important of these criteria. If the classifier is so fragile that minor changes to intensity or orientation of the image causes major changes in the classifier response, then such photometric invariances caused by the cameras would render the process unusable. Edge-based methods are robust against brightness changes on each image[112]. Selectivity is also a very important for the performance of the later stages of the process. If the classifier is too selective, then too few image features will be detected, and the later stages will have to interpolate between those points to regain structural detail. Interpolation is often very time consuming, and seldom accurate. If there are too many feature points detected, then the matching process which must decide which

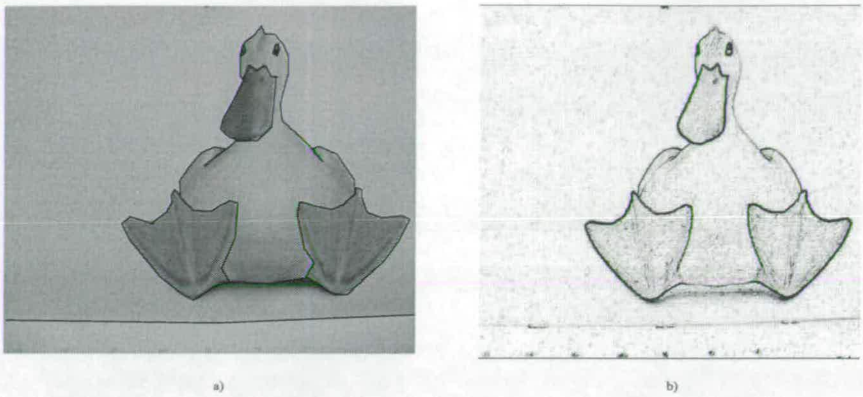


Figure 3.3: *Edge Detecting a simple object in a simple scene*

features in the left image match which in the right, will be swamped with candidates.

One of the earliest, and most respected of these feature detection algorithms is the edge-detector. It has been heavily researched, and thanks to the work of Marr, Hildreth, Canny, Deriche, Grimson and many others [37, 98, 105, 128–130], many excellent edge detectors exist for a variety of applications. Figure 3.3 a shows a simple image of a stuffed toy, with the important edges marked out. Figure 3.3b shows the image edge-detected using a sobel edge-detector. As can be seen, many edges have been detected within the image, including those that we marked as important.

Studies of the human eye have shown that although the eye contains over 100 million rods and cones, there are only 1.5 million fibres in the optic nerve, therefore certain operations must be performed on the images to compress the data while retaining the informational quality[10, 131, 132].

For bridging the gap between the computation theories and the biological data, we must first understand how the elementary computations are performed in neural hardware[125].

Figure 3.4 shows a simplified diagrammatic representation of the nerve layers within the retina. As can be seen there are three types visible. The Horizontal Cells connect neighbouring rods and cones together. They perform lateral inhibition, which enhances visual contrast. Wherever there is a rod excited by a strong light source, its neighbours (known as the surround), are

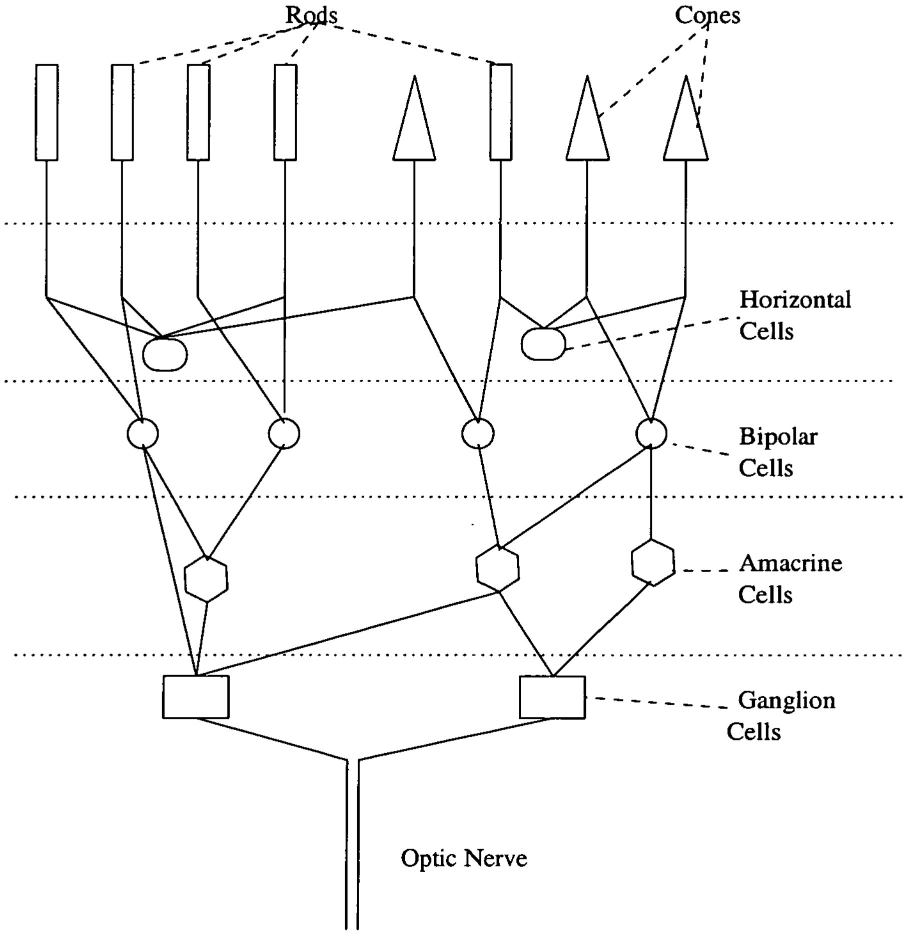


Figure 3.4: *A diagrammatic representation of the various nerve layers within the retina*

inhibited through the action of the horizontal cells. Coupled with another set of cells known as the Amacrine Cells, provide a very effective Gaussian filter in parallel across the entire image. The Bipolar Cells have varied functions. In certain cases they can be laterally inhibitory and in others they can be excitatory. This gives an acute method for separating contrast borders, i.e. a rudimentary form of edge detection[131].

The Ganglion Cells are the method by which the outputs of the Amacrine Cells are collated and transmitted to the visual Cortex. There are three distinct groups of ganglion cells, designated as W, X and Y cells. Each serves a different function. W cells account for approximately 40% of ganglion cells, and appear especially sensitive for detecting directional movement, and are probably important for much of our rod vision in the dark. The most numerous of the ganglion cells are the X cells, comprising 55% of the total. The X cells have narrow fields, and as such represent discrete areas within the visual field, and are probably responsible for colour vision. The Y ganglion cells have very broad reach throughout the retina, and respond to large changes in intensity. Therefore these ganglion cells respond to abnormal visual events but without great accuracy of location within the field. The ganglion cells also seem to perform some colour processing, therefore the process of colour analysis begins in the retina[131].

In conclusion, from inspection of the eye, it seems that there is much more going on than simply collecting light. It seems that before the information reaches the visual cortex, it has been smoothed, edge detected and segregated into movement-based, wide-area static and narrow-field static data. Using information such as this, researchers began to theorise about how the eye performs its job, and began to design systems based around their ideas.

After researching the massive compression that occurs between the recording surface of the retina, and the optic nerve ganglia, this researcher wonders if some time-multiplexed operation of the nerve system using buffers or 'taps', which does seem to fit with the layout of the visual cortex, may provide an alternative solution to the conundrum. As Koch states, information, is most likely coded in terms of the interval between successive action potentials[125]. There may therefore be evidence of frequency-based coding within the neurons, so therefore why not also between them? This, however, is topic enough for a whole different thesis.

In 1979, Hildreth and Marr produced a paper on the theory of edge detection[129]. Using the evidence given in neurophysiological papers, they show that using a Laplacian of a Gaussian operator is a near-optimal edge detector, and that it closely mirrors the processing done at a low level in the recording surface of the eye. Originally Marr[129], and later Laligant [133] discuss the variance in edge smoothness, due to lack of focusing, showing that a detector optimised for edges with an intensity profile which spans only a certain number of pixels, cannot pick up reliably all edges [129, 130, 133]. Coupled to their theories of five spatio-frequential "channels", Marr and Poggio developed a framework for an edge detector which would mimic the response found within the human eye, possibly providing a computational homologue of the compression employed.

Pixel intensity discontinuities, i.e. edges, are excellent features to match upon. Since they mark discontinuities, pairs of such edges can typically be used to find the boundaries of objects, and hence locate the objects themselves, and give interpolation routines consistent, usable anchors to work upon.

However, high-frequency detail and/or noise caused by lens problems, camera aberrations or electrical interference can cause edge detectors to mark edges at locations where no edge is present. The Marr-Hildreth edge detector marked edges at any zero-crossing output of a Laplacian of Gaussian filter. As Canny subsequently showed, this is not a practical proposition due to the high numbers of zero-crossings due to noise, even if the noise has vanishing energy. Canny proposed *Hysteresis*. Hysteresis uses two edge-detection passes across the image. The first is a coarse pass, and the second is a finer pass. Only the edges in the finer pass which also show up, at least in part, in the coarse pass are kept. While much slower than the single pass algorithm, Hysteresis produces much better results.

This shows that, as with many aspects of stereo vision, there is a trade-off. Some edge operators may find most edges but also respond to noise; others may be noise-insensitive, but miss crucial edges[34, 132].

Laligant et al use a system called Multiscale Merging to perform edge detection, which can detect edges of many thicknesses, which most edge detectors cannot, to overcome some of these problems[133].

Pivoting matching around information gleaned by edge-detection can cause problems because while edge-based methods are robust to photometric variance, they cannot be applied to objects with poor edges (e.g. human faces)[112]

While edge detection is an extremely attractive filter to the vision researcher, its use can lead to time and processing being spent correcting the possible problems caused by it. In fact, Rosin[75] opines that the process of extraction of low-level features followed by further processing based on these features accounts in part for the failure of generalised contemporary vision.

3.2.4 Corner Detection

Rosin's solution to this problem is to use *Corner Descriptors*. Rather than matching on simple single edges, junctions of simple edges are used. Corners are a popular low-level image feature used for matching in object recognition, stereo and motion analysis[75]. *Cornerity* can be quantified by a large number of parameters, improving the matching likelihood.

The successes shown in Rosin's paper and others, demonstrate that in the scenario where there are many corners within the scene to match, the algorithm performs extremely well. For this reason, the examples are typically office or part recognition scenes.

Unfortunately, in the cases where there are large numbers of corners, all with similar parameters, the algorithm begins to break down once again, in a similar manner to simple line-matching. An example of this would be a repetitive pattern. Also, because the system is still matching on features, the problems of occlusion and featurelessness are still present.

In conclusion, corner descriptors are a powerful and resilient method for generating a feature-based description of the scene, but despite this, the process is still feature-based, and therefore subject to the problems discussed earlier.

After examining all of these feature-based matching modules, it has become clear that a feature match-based system cannot function in a generalised manner without a large amount of *a priori* information about the scene, a knowledge base to guide the matching, or without a large post-processing module. Without these, the system will suffer from many false-positives, and will hence build erroneous depth maps. Hence, solving the SCP *without* requiring feature matching was investigated.

3.2.5 Model Matching

Commercially available applications to date have been mostly in automation of inspection and classification of objects in highly controlled settings[76]. Many authors circumvent problems by making hypotheses about the type of objects being observed[13]. One such way is to model each object that the system will see. Each object under scrutiny can then be matched against the internal model.

Model matching is a classification problem, and therefore is sometimes coupled with Neural Nets (*NNs*). The recent work by Zaki, for example, uses NN classifiers to attempt to identify occluded objects by matching them against known models[12].

Model matching has been used successfully in the part-selection and part- identification domains of robot vision. For recognising a certain object, one needs to compute some significant features and match them to those of a user-specified object model[44]. This can be very memory-intensive, and as the number of objects increases, the discrimination between various types of object becomes much more difficult.

One of the main problems in object recognition is the so-called "tolerance separability" problem. On the one hand, slight changes of the object's appearance should be tolerated, on the other hand, similar objects of different classes should be separated[100].

3.2.6 Shape Recognition

One of the most powerful subsystems of the human visual system, and one in which computers are still far behind humans is the Shape Recognition system [90]. Recovery of three dimensional information and shape recognition are two independent systems. Shapes can be recovered, for instance, just as easily from monocular views as binocular.

In order to measure shape, pixels which correspond to the same single object must be grouped together in a robust manner. This process is one of the major problems in the area of correlation-based stereo matching [21, 54]. In images with low contrast, it is a common error to group feature points together which are not part of the same object. Region- growing is a popular solution to this however it involves large amounts of memory, and can be a rather slow implementation. By inferring some information about pixel grouping globally, from patterns within the intensity information, one can generate far more reliable results than simple local-matching rules provided by region-growing.

Many shape recognition algorithms come from the branch of the field commonly known as Shape-from-X. Grimson[97] did a lot of work on reconstructing surfaces using binocular shading. In his conclusion he states that despite the problems with detecting known anchors, such as lines, because they provide directional information at discontinuities, it is still possible to use such information to reconstruct surfaces reliably, however the numerical stability of the computations degraded rapidly when less was known about the image, such as light source positioning[20,97]. It is very difficult to take into consideration the effects of inter-reflections, specularity, and innumerable other non-linear effects in a general situation - since that requires a prior knowledge of the object geometry itself![72]

Because of these problems, and the issue of run-time, this visual module was not used within the project.

3.2.7 Graphs and Flow Networks

Flow networks are typically used to model the flow of material (or data) from one location to another through an interconnected network of channels. Flow Networks consist of a number of Vertices (V) and a number of Edges (E) which connect them. Each edge has a nonnegative **capacity** $c(u, v) \geq 0$. If $(u, v) \notin E$, i.e., there is not an edge between the vertices u and v, then the capacity $C(u, v)$ is assumed to be 0. We distinguish two vertices in a flow network: There is a **source** (s) and a **sink** (k), between which the material flows. A flow Network can therefore be defined as $G = (V, E)$, and for convenience, we can assume that every vertex in V lies on some path between s and k.[134]

In Figure 3.5 the times taken to travel between various locations in Edinburgh are shown. One can use this to trace the fastest route to the sink vertex "Office" from the source "Home".

In this example, the network is merely a Directed Acyclic Graph with edge weights. However, the traversal of material (i.e.. the traveller) from one location to another is represented.

By using flow networks in this way, we can model an **optimum path** through a network, rather than **optimum flow**.

The product of the first stage of the project, which will be described in Chapter Four, is a large matrix of values. Through this matrix, there is some path from one corner to the opposite corner for which the values of the data points it passes through is a maximum. It is towards this task that the directed acyclic graphs and flow networks are brought to bear.

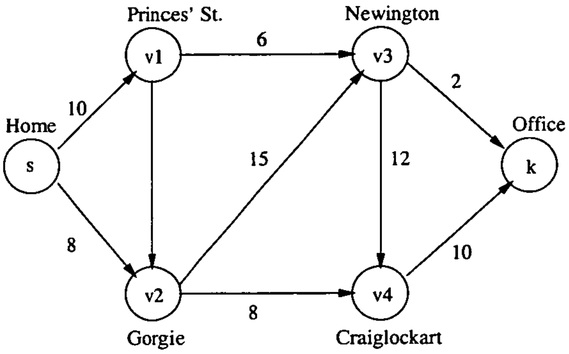


Figure 3.5: *An Example of a Directed Graph*

3.2.8 Dynamic Programming

Dynamic Programming (DP) is an efficient method of optimisation for functions with many discrete variables. In this way stereo correspondence is fairly well suited for DP[7, 43, 52].

Fielding states that "while a Dijkstra-type shortest-path algorithm, solves the problem, a more efficient algorithm is possible [using DP] because of the regular structure of the graph" [25].

DP was considered for use in this task. However it was tested, and there seem to be too many differing constraints on the problem, namely the uniqueness and smoothness constraints, and the ability for the solution to contain gaps of unspecified size and quality. It was also noted that a configurable, hierarchical method was needed, which proved too difficult to incorporate into the DP model.

When an initial DP algorithm was employed, a further flow-step was required to find the cost of 'jumping' over less-suited areas, and the super-network was still required to join all of the smaller paths together.

Because of these often conflicting constraints, graphs and flows offered a more robust and malleable solution. Dynamic Programming especially has problems in cases where there are several maxima[25, 135], differing only in their results from the smoothness constraint. This gave unacceptably poor results, especially in cases where the camera noise was obtrusive.

Fielding admits that, in unstructured environments where narrow occluding objects may be present, maximum weighted and greedy matching are preferred over dynamic programming[25].

3.3 Spatial Approaches

Having reviewed each of the major approaches to stereo vision, and taken on board the comments of the researchers over the last ten years, it has become clear that with modern technology, at the present moment, spatial "pixel-based" stereo seem to be the most promising to yield a testable solution.

The reviews below concern work done by various researchers over a forty year span. Since much of this research was originally done in secret, few of the researchers used the same datasets. Also, much of the research was done by Artificial Intelligence experts, while other research stemmed from the fields of Computational Science, and Electronic Theory. Therefore, few of the papers use the same metrics. This renders any comparison subjective, and unsatisfactory for a comprehensive Thesis. Therefore, where comparison is difficult, the approximate error from the required results (usually depth or disparity maps) has been taken and evaluated for quality. Where such data is not given, evaluations from later summary papers are used.

The first method described is that of Marr and Poggio's Cooperative Stereo Algorithm, which forms the basis for this project.

3.3.1 The Cooperative Stereo Algorithm

In 1976, Marr and Poggio described a *cooperative* algorithm that solves the correspondence problem for stereopsis[36]. They argued that the stereo problem could be reduced to that of matching two primitive descriptions, one from each eye, the central problem being to find a correspondence between those descriptions that satisfy the Uniqueness and Continuity Constraints[61].

The term "cooperative" refers to the way in which local operations appear to cooperate in forming global order in a well regulated manner[36]. The name came from Julesz' original work into Cyclopean Perception and Random Dot Stereograms in 1960. Julesz believed that stereoscopy was a *cooperative* process[1, 136, 137].

For example, in the eye, the neural connections between the rods and cones cooperate in parallel to perform smoothing, enhancement and compression on the image before it reaches the optic nerve. There are many such examples of cooperation in nature [36].

Marr and Poggio devised a way of representing the stereo matching process, based upon earlier work by Julesz who used a matrix of dipoles. In Marr and Poggio's system, the epipolar lines to be matched are laid out along the X and Y axes of a grid. Each cell, or node, at position (x,y) corresponds to the possible match between the xth pixel of the left epipolar line against the yth pixel of the right epipolar line [1]. Figure 3.6 shows the local structure around a node.

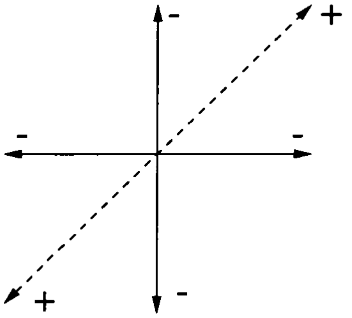


Figure 3.6: *The Local Structure around a single Node in the cooperative network.*

The Uniqueness constraint is enforced by allowing only one match on any one column or row. Therefore, no pixel in the left can match more than one pixel in the right, and vice versa. This is enforced by having inhibitory connections along the rows and columns.

The Continuity Constraint is enforced by giving excitatory connections along the diagonals between cells. In this way, it is more likely for a match to occur in the vicinity of another match. Figure 3.7 shows the explicit structure of the two constraints for the case of one-dimensional images (or the matching step for one epipolar line from two two-dimensional images). Solid lines represent "inhibitory" interactions, while dotted lines represent excitatory ones.

The epipolar lines are placed along the axes of the node grid, and each node(x, y) is given a value of "on" or "off" depending upon whether the epipolar lines have matching pixels at the coordinates x_l and y_r. The excitatory and inhibitory influences are then calculated. Any node whose value is increased above a threshold level by the excitatory influences over it, is set to "on". Similarly any node whose value is decreased below a threshold is set "off". The network will stabilise into a state whereby clusters of nodes will be "on", and the disparity can be simply read off from those nodes.

This inhibitory and excitatory connection scheme between cells has been likened to the Game of Life[1], in the way that the status of cells can affect the cells around it.

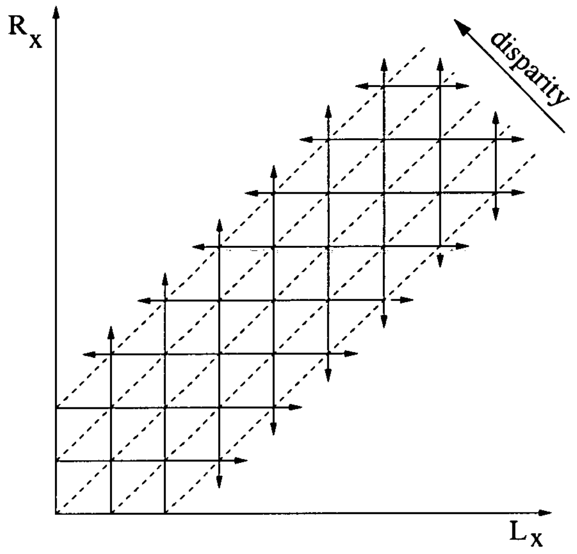


Figure 3.7: *The structure of the cooperative network for a one dimensional image.*

Originally, Marr and Poggio used their algorithm against Random Dot Stereograms, and obtained very impressive results. They were unable, at the time, to gain a general theory of how the algorithm would perform in real life, until their paper a year later, in which they conclude that "It seems unlikely that one can construct a useful general theory of the [Random Dot Stereogram] Algorithm" [61].

Indeed, they are thorough in attempting to model the random process by which the stereograms are created, and show clearly how their algorithm begins to degrade with the sparcity of the data. They also theorise about how to modify their algorithm, however they do not implement such, to attempt to deal with noisy or non-planar data. Since the clustering that is occurring due to the diagonal excitation tends to form clusters at single disparities.[61] Any real world algorithm based upon their work, must, therefore (as well as deal with real world data, not random-dot stereograms), overcome the planar and noise problems.

This method of recovering depth is extremely attractive for a number of reasons. Firstly, the simplicity of the solution, and the core algorithm which utilises linear memory searches lends itself to high optimisation, and is a most likely candidate for embedding within a VLSI. Secondly, much of the work can be subdivided and handled in parallel, with the splitting of source data and recombining of results trivial; something which many of the frequential methods would find difficult to implement on general purpose hardware. Thirdly, the algorithm itself encapsulates neatly and elegantly the two central constraints of the field, requiring no successive pruning steps in that regard. Lastly, as will be shown in the following chapters, the algorithm is highly adaptable. The central inhibitory/excitatory conditions can be modified to accept non-binary intensity data and use hysteresis.

3.3.2 Sum of Sum of Squared Differences

Two of the most well known researchers in the field of conventional stereo are Takeo Kanade and Masatoshi Okutomi.

Their work was designed to solve a paradox outlined earlier in this Thesis. In stereo processing, the further apart one places the cameras the more precise the estimated depth. However, there is much more space that much be searched, and the possibility of finding a false match is much greater, so there is a trade-off between precision and accuracy in matching[3].

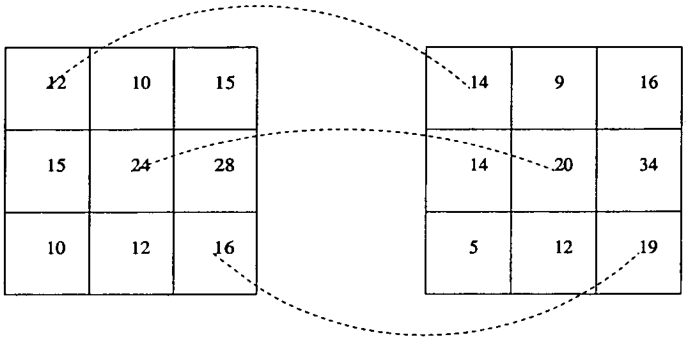
In performing stereo searching, simple pixel comparisons are inadequate, since the possibility of false matches are too high, therefore one must take regions of pixels and compare them. The most common method for comparing these areas is taking *Sum of Squared Differences* (SSD) between them. This method produces dense disparity maps, a very desirable result[70]. A slightly more computation-friendly block differencing metric is the *Sum of Absolute Differences* or SAD[49, 50, 115, 126, 135]. If a candidate window in the second image exactly matches the reference window in the first, then the SAD of the two windows becomes 0[49]. As an example of SAD in practice, Ninomiya uses SAD in his motion-estimation system for vehicular control[115], and reports good matching.

SAD and SSD are usually justified on the grounds that they are easy to implement and use less computing power[135]. Work by Corke and Sun, however, shows that both SSD and SAD are not robust with respect to radiometric distortion[118, 135], a problem with most off-the-shelf cameras[46]. This distortion is usually non-linear, and therefore hard to model[46], and therefore most existing camera calibration techniques are based on the pin-hole camera model[78].

To counter this, they propose a Zero Mean Normalised Cross Correlation (ZNCC), a metric with much higher computational overhead (including a square-root term). They circumvent the overhead problems using custom hardware.

Figure 3.8 shows two 3x3 blocks of pixels. Each of these blocks corresponds to the image cast onto the recording surface of two cameras viewing an office scene stereoscopically. Equation

3.3.2 shows how the SSD is calculated by subtracting the values in the corresponding pixels, squaring the result and summing overall.



12	10	15
15	24	28
10	12	16

14	9	16
14	20	34
5	12	19

Figure 3.8: *Pixel areas from two images from a stereoscopic pair*

$$\begin{aligned} \text{SSD} = \quad & (12-14)^2 + (10-9)^2 + (15-16)^2 + (15-14)^2 + (24-20)^2 + \\ & (28-34)^2 + (10-5)^2 + (12-12)^2 + (16-19)^2 \\ = \quad & 4+1+1+1+16+36+25+0+9 \\ = \quad & 93 \end{aligned}$$

This method is effective, simple to implement, and is in use in a number of stereo systems[3, 30, 70]. However, the central problem with SSD is finding an appropriate window size. As was discussed earlier, if the block size is too large, the computational burden becomes large, and smaller features tend to become smoothed over. If the block size is too small, the possibility of false positives increases, and the result is a poor disparity estimate[70].

Kanade and Okutomi provided two possible methods for overcoming some of the false matching problems. Both methods were released in papers in 1990. The first, that of using an Adaptive Window Technique uses a very greedy gradient descent algorithm[138]. It is their second method, *Multiple-Baseline Stereo*, which will be discussed.

Work by Sahabi and Basu in 1995 shows that the error expected at various depths within a scene can be analysed and reduced by using vergence and varying the geometry of the cameras[18]. The complexity introduced when the cameras are allowed a variable geometry that allows vergence is often undesirable, however. Kanade and Okutomi tried to overcome the problem of different error estimations at different depths, however, keeping the cameras coplanar.

Their method used multiple stereo pairs with different baselines generated by a lateral displacement of a camera. Figure 3.9 shows a typical arrangement of three cameras in an SSD rig. The cameras can be paired up to produce three stereo rigs of varying baselines, 4, 6 and 10 units.

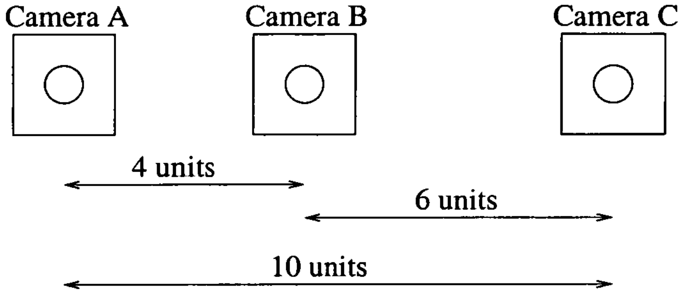


Figure 3.9: *Three cameras in an SSD Rig*

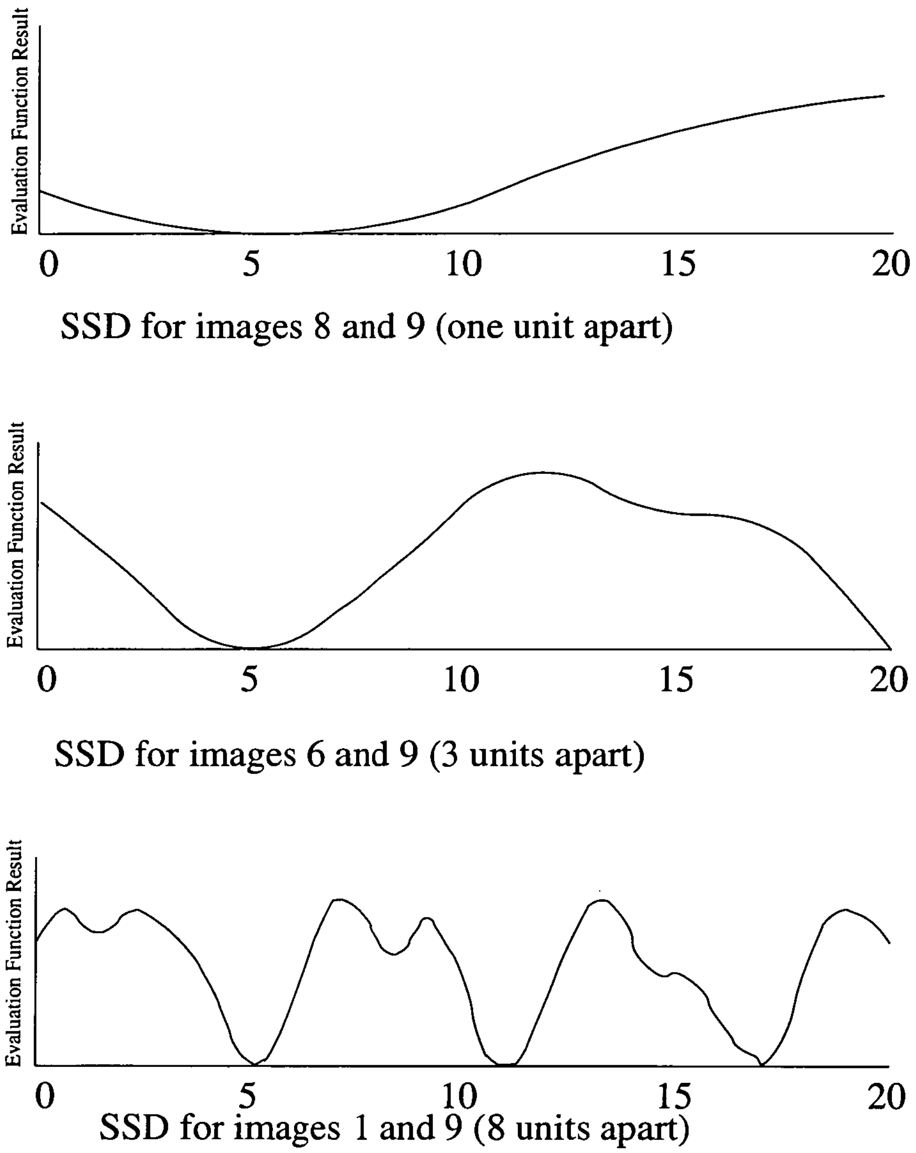
This deployment of the third camera is useful for standard stereo reconstruction, as is provides a method for evidence when a match is made. However, Kanade and Okutomi took this theory one step further. They suggested using cameras A and B together with an SSD depth discovery algorithm, and Cameras B and C similarly. By using knowledge of the geometry of the cameras to allow the results to be plotted together onto the same graph, and then summing the results, one can get a Sum of Sum of Squared Differences Estimate of the depth (SSSD)[70].

In the 1990 paper, Okutomi and Kanade use 10 separate images, simulating 10 cameras displaced laterally to view a town scene. The distance between each camera is 1.2cm, and the depth to the object is approximately 1m.

They then showed the SSD results from pairing up each of the images in increasing baseline. To illustrate the SSD algorithm, Figure 3.10 shows the disparity estimates from a particular location 'x' within a repetitive area of the images.

The images 8 and 9 are only one 'separation unit' apart (i.e. 1.2cm), and have poor disparity differentiation, however there are few minima. Images 1 and 9, however, which are 8 units apart (i.e. 9.6cm), have a very accurate depth measurement, but there are three minima within the 20-pixel search window. If one sums these three graphs together, the graph shown in Figure 3.11 is the resultant disparity estimation. As can be seen, the correct depth is read off from the value with the minimum error. In this case, the point 'x' has a disparity value of 5.

As more of these multi-baseline results are combined, the function becomes more selective, and gives rise to what Okutomi calls "SSSD-in-inverse-depth". Unfortunately, as the number of cameras increases, so does the complexity.



SSD for images 8 and 9 (one unit apart)

SSD for images 6 and 9 (3 units apart)

SSD for images 1 and 9 (8 units apart)

Figure 3.10: *SSD results for one epipolar line of 3 images. From Okutomi and Kanade[3]*

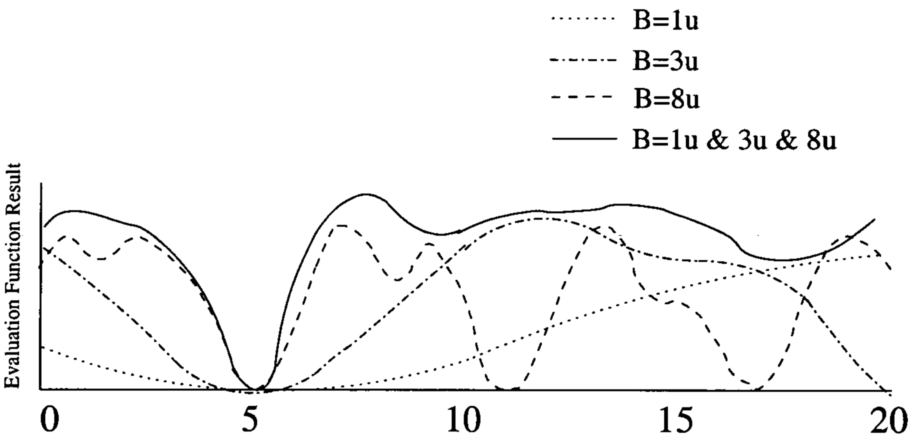


Figure 3.11: *SSSD of the three graphs shown in Figure 3.10. From Okutomi and Kanade[3]*

In a later paper, Dhond and Aggarwal found that the increase in computation due to a third camera was only 25% while the decrease in false matches was greater than 50%[30, 111].

This researcher has not found any empirical evidence in any of the papers to suggest a point past which the increased computation of adding further cameras outweighs the improvement in results in the SSSD environment. One must, however, consider the practicalities in keeping up to nine cameras aligned, coplanar, and clean. In my experiments, I found it hard to keep two cameras aligned, even using custom-designed brackets.

In later work by Ross at CMU, a three camera stereo rig was employed, and the SSSD system implemented for use with mobile robotics. The implementation is a great improvement on the work by Okutomi and Kanade. The system is optimised in a number of areas, and although the large baseline (1 metre), means that the search window is 120 pixels, rather than 20 as given in the earlier work, Ross uses some adaptive techniques to reduce this to about 50 in some areas while avoiding the slower adaptive windowing techniques. He reports very fast solution times, using low resolution images, and good results, with problems arising only in areas with few features and little surface texture[30]. This, however, is a problem with almost all stereo matching systems.

In later work, still, Kanade et al, create a very fast Depth Map generating machine, using the SSSD algorithm they outlined in their paper six years previously. In order to achieve the improvements necessary to gain a video-rate system they had to make a number of modifications, since they estimated that 6 cameras producing 256x256 images, with a search window size of only 11 with the required frame rate of 30 frames per second to be 465 Giga operations per second[71].

- A specialist machine was created.
- An SSAD subsystem was employed instead of SSSD.
- Small images of size 200x200 were used.

It seems conclusive, therefore, that the SSSD method is both simple, extensible, easily implemented in parallel hardware, robust and reliable in most cases. However its reliance on multiple cameras, all exactly aligned, and with larger than usual baselines means that it has limited use in most situations, and cannot be used to solve the *binocular* stereo correspondence problem, side-stepping the problem with the use of more cameras. Also, as can be seen from the various implementations, the number of operations is so high, that it is unlikely that a general purpose machine of over 465GHz will be implemented soon, therefore specialist parallel pipelined hardware will be needed for the foreseeable future.

3.3.3 Using the DCT to make a better Edge Matcher

As with all things in Computer Vision, there are ways to combine approaches. This final investigation of conventional methods shows how a conventional framework can be used with frequential components.

Based at Essex University, D.V. Papadimitriou and T.J. Dennis are well known in the Computer Vision field, and have produced some excellent work in the use of frequential methods to solve the SCP[139] Another of their methods is discussed in the Section on Frequential Approaches, below.

Pagliari, working with Dennis, constructed an edge matcher using a Discrete Cosine Transform.[140] The main advantage is that the 2 dimensional DCT gives positional *and* directional information of the classified image points, which greatly reduces the searching in later stages.

The main problem that this system was designed to overcome was that traditional spatial edge detectors can be mislead by factors such as non-constant illumination, occlusion and depth discontinuity ambiguities. Their algorithm tackles these problems by dividing the images into 4x4 blocks and dividing them into high detail and low detail blocks. The high detail blocks are then classified by performing a 4x4 DCT upon them, and then classified into one of the 16, 4x4

DCT bases, and then sub classified into one of eight edge orientations. Figure 3.12 shows the 64 DCT Base functions for an 8x8 block. Any grey-scale 8x8 pixel block can be fully represented by a weighted sum of these 64 basis functions with the DCT coefficients acting as weights for that block.

The initial results of the edge matcher are very good. In order to test their edge-classifier, Pagliari and Dennis construct a subsequent simple 2 dimensional search routine such that for each image in a high-detail block, a 5x5 window in the other image is searched for matches. This generates another coarse-to-fine search method, albeit one with only two layers. Their solution for matching low-detail blocks is to region-grow around them until a high-detail "anchor" block is found.

Their experimental results, using a head and shoulders image, demonstrate the operation of the DCT edge classifier and generate a satisfactory depth map. The only points of failure were due to the well known problems of occlusion and featurelessness. This, however was due to their testbed system, and not the DCT edge classifier.

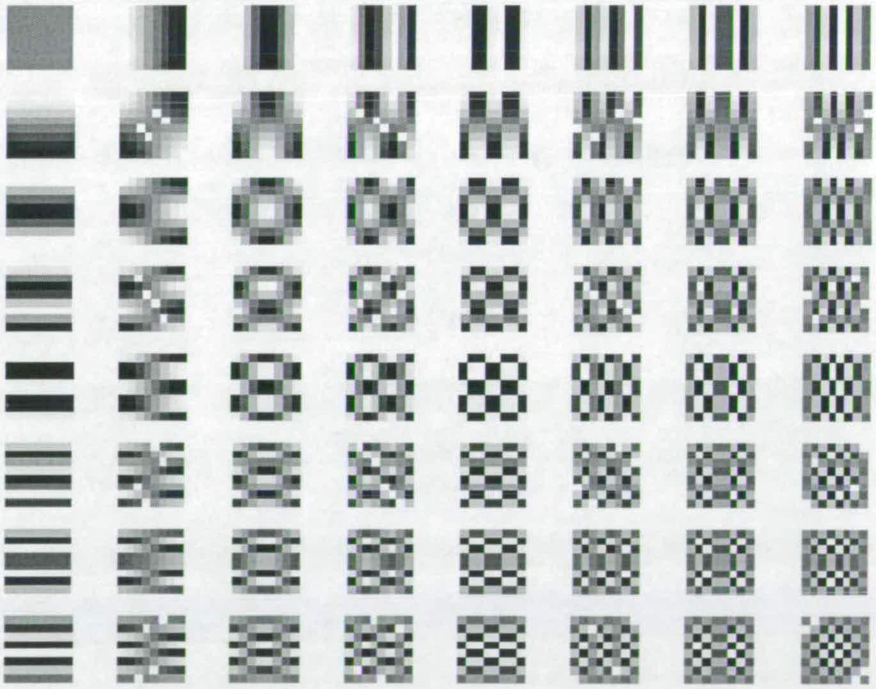


Figure 3.12: *The 64, 8x8 DCT Basis functions*

3.4 Frequential Approaches

Marr and Poggio very much led the frequential stereo camp, in a reversal of their recent previous work in cooperative stereo. Their cooperative algorithm, still had flaws. The algorithm did not work too well on natural data, and the research being done at the time in the field of neurophysiology, on the workings of the ganglionic connections within the retina, was beginning to shed light on the possible workings of the human stereo fusion system. Some of these findings seemed to disprove some of Julesz' work.

Like feature-based methods which rely on edge-detection, phase-based methods are only accurate at edges[82]. Because they typically only match edges, they also produce sparse depth maps[82].

3.4.1 Marr's "Theory of Human Stereo Vision"

As new evidence for the*frequential* nature of the workings of the eye came to light, Marr and Poggio saw the possibility of a solution to a number of the problems they were running into in their cooperative stereo work.

Some of the new findings even cast doubt on the relevance of cooperative algorithms to the question of the fusion process in human stereo vision.[1]

Findings concerning independent spatial-frequency-tuned channels and the physiological, clinical and psychophysical evidence about the three-pool hypothesis[1], caused great interest.

The human visual system possesses both range and resolution, properties that a system comprising only three disparity levels would not provide. The existence of independent spatial-frequency-tuned channels suggested a more elegant two-stage scheme for solving the fusion problem.

1. Each image is analysed through channels of various coarseness, and matching takes place between corresponding channels from the two eyes.
2. Coarse channels control the eye vergence movements causing finer channels to come into correspondence.

[1] A theory attributed to W. Richards whereby there are only three disparity pools used in the disparity system; crossed, uncrossed and zero disparity

This scheme provides a completely separate solution compared with Julesz' work, as it contains no hysteresis, a key observation of Julesz'. To explain this, Marr combines the tuned channels with a system outlined in a paper with himself and Nishihara, in which they suggested that there was an intermediate step between the early vision and the final fusion, the $2\frac{1}{2}$d sketch. This sketch, they say, acts as a *memory buffer*, and may provide a solution to the lack of hysteresis. They incorporate this theory of the $2\frac{1}{2}$d sketch by adding two more steps to their scheme given above.

1. When a correspondence is achieved, it is held and written down in the $2\frac{1}{2}$d sketch.
2. There is a backwards relation between the memory and the masks, perhaps by the use of eye movements that facilitate surface fusion.

This system provides a very powerful solution to the stereo problem. In the conclusion, it is admitted that there are a number of points which are still uncertain. The structure of the $2\frac{1}{2}$d sketch is not given, nor are any hypotheses about how the vergence of the eyes may be controlled. However, to reproduce such a system within a computer, these are not difficult problems. One can structure a computer homologue of the $2\frac{1}{2}$d sketch in any way necessary, and the problem of Panum's Fusional area does not apply to a computer where resolution of the image is uniform, and not concentrated in the fovea.

In 1980, Grimson created just such a system. In accordance with the psychophysical data obtained, he used four $\nabla^2 G$ operators, with widths of 9,18,36 and 72 pixels. He tested it on stereograms as well as natural data. Qualitative evaluation was difficult to obtain for the natural images as the imaging geometry was not controlled, but from observation the algorithm's performance was still good[37, 141].

The algorithm was revised and enhanced, and in 1984, Grimson released another paper on the subject taking into account these latest advances. He reported that a single stereo pair can be fused within tolerance, using three channels, on a special-purpose LISP machine in approximately 10 minutes.

The various systems work very well, and the error rates given are very impressive indeed. The concept of using data passed through a number of different frequential channels provides a very useful way of gaining evidence when a match is hypothesised, and zeroing in upon it, to gain more accurate results.

Gaussian smoothing the images, the concept of hysteresis, and using a coarse channel to guide a finer channel is very attractive, and much research was done into incorporating this into the system. Unfortunately, as expected, the computational overhead is too high to create a fully hybrid system. However, the benefits given by using a coarse channel to guide, and affirm, matching far outweighed the overheads, and therefore a simplified smoothing operator is used in the "aggregation" step, as will be explained later.

The field of Frequential Stereo research did not end with Grimson's review. Indeed many researchers have used unconventional methods in an attempt to solve the SCP.

3.4.2 Jepson and Jenkin's Fast Phase Difference Computation Algorithm

In 1987, Jenkin, Jepson and Tsotsos, in a University of Toronto internal report, worked out an algorithm for implementation on an analog machine. They stated that many of the problems encountered in the measurement of disparity could be avoided by treating the measurement of disparity from bandpass signals as being the equivalent to measuring the local phase difference between them[29]. Unfortunately their algorithm, while producing some promising results for the time, was very inefficient to implement digitally.

By 1989, many researchers were using the advances in computation to turn their hypotheses into working systems. However, the systems, (Jepson et al's included), used their frequential power in a 'brute force' manner, by solving differential equations at a large number of image points[29].

Earlier in 1984, Nishihara proposed a system of coarse to fine stereopsis which might circumvent much of the brute-force application of 'heavy' mathematical computation[142], extending some of the earlier hypotheses of Marr and Poggio. Nishihara's hypothesis was that doing a few rough estimates at a coarse matching level could be used to guide a finer matcher, reducing the work required.

Jepson and Jenkin saw potential in Nishihara's work, and combined it with their own earlier technique to produce a system which computed disparity using phase differences, much faster than their earlier attempts.

The main advantage to their work was that, unlike Feature-based matching where specific feature points, such as zero-crossings are used to correlate matching, the entire content of the signal can be examined, resulting in denser depth maps.

One of the problems with the method was that unless the original images were brought into close alignment, the phase differences were too great resulting in a lack of match, or an infeasible computational overhead. Once again, the fusion of their work and Nishihara's provided a solution.

They found that even a coarse match in the frequential domain provided a fair disparity measurement. Therefore by performing a rough match first, the results could be used to bring the images into closer alignment for the finer matches without searching for match tokens, as in a classic coarse-to-fine stereopsis algorithm.

They use "demons", which are separate differential equation solvers. Each demon is presented with two sections of data, and requested to report the strength of any "locks" found for a given coarseness and offset. The offset is determined by a demon from a higher coarseness layer of the pyramid for the same location in the image. Any demons which do not lock are given new starting offsets, gleaned from any demons in the neighbourhood which have locked. If the demon still cannot lock, it is marked as "non-locking". All detector demon outputs are averaged with their neighbours.

Pyramid layers are separated one octave in scale. When each layer is completed, the next finer layer is begun, with its initial offsets being given by the recently completed layer. This continues until the finest layer is completed.

Jepson et al discovered that additional constraints were required, and that they parallelled closely earlier work by Pollard, Mayhew and Frisby. They suggested that if the disparity gradient was greater than a certain threshold, then there was probably something wrong with the match, and therefore it could not be relied upon. Jepson found that due to the bandpass nature of the data, the phases can only vary within a set rate, so if the demon lock results varied too greatly, then the results must be invalid, and the certainty factor was set to zero. Each offending area was given a disparity as an average of its neighbours. This notion of a "disparity gradient" was incorporated into an algorithm by Pollard et al[143]. Pollard's reasoning, however, was based upon psychophysical evidence.

Jepson tested his algorithm on Random Dot Stereograms, as well as real-world stereo image data. The results for the Stereograms are fair, however the author admits to setting the dot size to correspond to the wavelength of the finest tier of the pyramid. When this coincidence is removed, the algorithm begins to degrade, but still manages to recover some depth information. On the real world data, the algorithm performs similarly. The automobile-part image provides the best results due to the good overall quality of the image. The "Pentagon" image, also provides a fair height plot, with the building clearly discernible from the ground surrounding it.

Their method is sound, and as they mentioned in their discussion, supported by psychophysical evidence. The combination of their earlier work with Nishihara's and Pollard et al's is elegant, and shows that frequential solution can be optimised for digital hardware, and produce fair results, even without any monocular clues.

3.5 Connectionist Approaches

The SCP is a hard problem from a computing point of view because there is a large volume of data pouring in each second which must be matched to another large volume of data. However the rules for the matching keep changing, are sometimes unknown, and often have no way to check whether they are correct or not without human intervention.

Attempting to achieve human-like performance, many researchers have been using NNs to solve the matching problem based on binocular images[102]. NNs are ideal for solving problems whereby one has a small amount of representative data, but must to use it on a large amount of unknown data[144].

Neural Networks function as large Black Box Learning Machines. Once a NN is trained, even the researcher who designed it will have a hard time deciphering the decision methods the NN uses. This is because a NN consists of a large number of small processors, or Neurons (or Perceptrons)[85, 86]. Each Neuron is responsible for a tiny fraction of the problem, and may, in itself, not even solve the whole of that. Each Neuron may, in fact, be part of the solution to two or three different problems. Despite this, they are useful because they can often learn to solve problems that cannot be solved easily using mathematics. The SCP is just such a problem.

Therefore, one has to make a trade-off. If one is willing to forego the knowledge of how the SCP is solved, a large NN can be implemented to solve it. However, NN's are still not guaranteed to produce an accurate solution each time. A Stereo Vision system will 'see' a lot of things during its lifetime, and it is possible that one situation will not have been covered by the initial test data. NN training takes some time, so if one allows the NN to train itself on the data coming in the process will need to be guided (or else the system will train itself erroneously eventually), negating the usefulness of a fully automatic vision system, and will probably slow down the matching process, also negating the speed advantages of a NN in the first place.

3.5.1 Nichani's Hopfield Net

A Hopfield Network consists of a set of neurons, where the output of every neuron is fed back into the input of every other neuron. The weights of these connections are fixed, and are constrained to be symmetrical[26]. The network (hopefully) settles or *relaxes* into a stable state corresponding to the solution. The state of the network as a whole can be considered as an Energy Function. Over time, the links will change, and the metric used will evaluate that state into a value for the energy function. Over time, this function will decrease, finally reaching a local minimum.

Hopfield Networks are used to solve difficult optimisation problems. The features that are used to establish the correspondences are edge points. The problem is first formulated as an optimisation problem, where an energy function is to be minimised. This optimisation problem is then solved using a Hopfield Network.

The technique can also be considered as a constraint satisfaction process, where each node corresponds to one possible match between two features, and the links are the compatibility of the correspondences, which are derived by making assumptions about the objects being imaged. [145].

Stereo Correspondence can be considered as an optimisation or constraint satisfaction process[9, 26], and constraint satisfaction can be considered an energy minimisation process[55]. Venkatesh, for instance, casts the problem of contour modelling as just such an energy minimisation process[146].

For example, a neuron n_{ij} explores the hypothesis that edge $_i$ in the first image corresponds to the edge $_j$ in the second image. If we have two neurons $\mathrm{n}_{i_1 j_1}$ and $\mathrm{n}_{i_2 j_2}$ which correspond to the likelihood of edge $_{i_1}$ matching edge $_{j_1}$ and similarly edge i_2 matching edge j_2, then the connection between the two neurons $(\mathrm{W}_{i_1 j_1 i_2 j_2})$ represents the compatibility of the two correspondences. This example is shown in Figure 3.13.

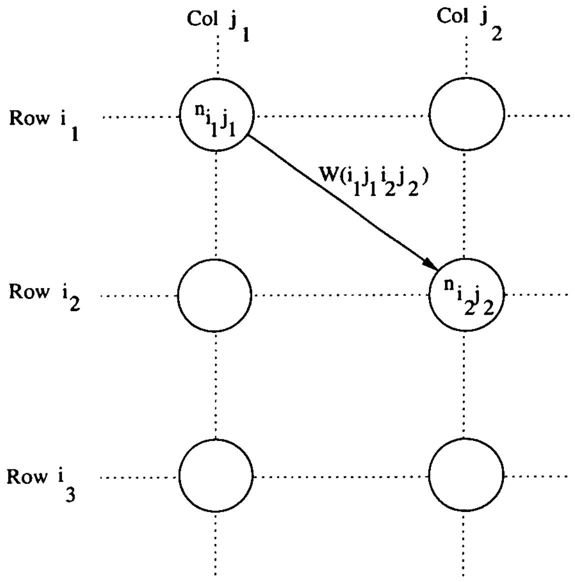


Figure 3.13: *A Hopfield Net*

For example if the first match disproves the validity of the second match, the connection will be inhibitory. If the first match proves, or assists the second match, the connection will be excitatory. As usual, the connection weights are computed according to the uniqueness and smoothness constraints.

In their favour, Hopfield Networks are robust, well understood, well behaved networks, which need little or no training. Once the network has settled down, it is a trivial task to read off the link weights to find a set of compatible matches.

However, they are heavily reliant on two things. Firstly, the designer must use a good quality energy function. Since the energy function can only be hypothesised, this is extremely difficult, and usually relies on various constraints being met. Secondly, they rely on feature extraction tools, such as edge detectors to pick out the points with which to match the nodes. As has been discussed in Chapter Two, in low light or poor camera conditions, or even in situations where there are too many edges, or fuzzy edges, the network can be supplied with either too many, or (as is often the case) too few edges because of occlusion.

The algorithm described in the paper by Nichani [145] is very typical of this sort of solution to the SCP, which is why it was chosen as a representative paper.

Nichani follows this standard layout for Hopfield networks, laying out the neurons in a two dimensional lattice. Nichani reports that the network, once started, converged rapidly to a good solution. His data was the standard "Pentagon" data set. The edge-maps displayed are fair for the dataset, giving good-quality edges with little noise. Familiarity with this dataset allows one to speculate that the edge- detector's threshold was set manually, and exactly to generate good edge- maps.

Nichani does build in the uniqueness and smoothness constraints into his solution using the notion of similar *disparity vectors* i.e. nearby neurons will have similar disparity values. Again, knowledge of his dataset tells us that this will work very well in the Pentagon dataset, but in an image where there are many objects at wildly varying disparities (such as a natural scene), this approach to the smoothness constraint could cause some problems.

It was decided to test just such a constraint, and a notion of "disparity vectors" (although the vectors are all one dimensional, thanks to the epipolar constraint in this project), was built in. It was found that in all cases of occlusion, or where disparity varied greater than some threshold value, the constraint caused difficulties. In some cases, however, it proved very useful, therefore a very weakened version of the constraint was left in the network reduction module, which allowed the constraint to *decouple* areas of image, if the disparity between them was too large (a threshold of 3 disparity levels).

However, Nichani gives some qualitative results of his method, and shows that, although crude, the system can work. To build a dense depth map, a further interpolation step would be necessary afterwards, however.

3.5.2 Koch's Analog Networks

Hopfield Networks generate "Binary" solutions. By this it is meant that the result is generally a yes/no, or a string of "yes/no" sub-decisions. In the case of edge-classification, this is highly desirable. When smooth surfaces need to be reconstructed, binary responses are insufficient for this type of classification. To reconstruct continuous, *smooth* surfaces, one requires analog output. For this purpose, one can use an Analog "Neurone" Net. Grimson's work attempted surface interpolation using zero-crossings as features[147], based on earlier work by Poggio and Marr.

A *classifier* is a simplified term for a neural net. It takes a set of inputs and classifies them into one or more categories. Whereas Hopfield Networks are fully interconnected, traditional

Neural Nets are not. They consist of a set of *input nodes*, which are set to the values of the input data, an *output node*, which will assume the value of the 'answer', and a set of *hidden layers* which perform the working. This can be seen more clearly in figure 3.14. The number of hidden layers indicates the maximum complexity of the model to be fitted. For example, if there are no hidden layers, the classifier can deal with linear solutions, a so-called *Linear Classifier*. As one increases the number of hidden layers, the system can 'learn' more complex systems, up to multi-dimensional ones, which require three or more hidden layers.

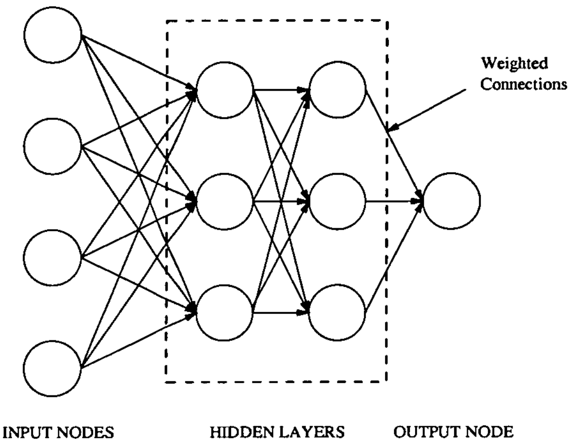


Figure 3.14: *A Neural Net (or Classifier)*

In a later paper, Koch et al create an Analog "Neurone" Net which solves the SCP in simple cases[92]. Their initial test data sets are Random-Dot Stereograms. They express the surface reconstruction as the minimisation of a quadratic energy function, and construct a series of equations to explain it. They use a lattice of analog circuit components to explain the scheme, using ohmic resistances and digital switches, which they claim use the best of both worlds. They do, however admit that their equations tend to smooth over edges. Therefore, they incorporate work by several other researchers, increase the number of terms, and therefore introduce a non-linear element, allowing discontinuities.

They test their network by simulating the analog network on a digital computer. The system is run over a 32x32 pixel image. They found that run-time was very dependent upon the weights given within the lattice. Setting these values had many side-effects, and had to be chosen carefully. They show how varying certain elements within the equations changes the convergence time and results, and demonstrate that qualitatively, the results are "Very good". Run-times are given in terms of multiples of "time constants", since the computer is simulating an analog network. They estimate, however, that if the hardware was built from semiconductor components, that the convergence time would be between 10 and 100nanoseconds, an impressive result. By giving this, however, they do side-step the issue of NN run-time on the general digital computer.

Their results are very heartening for the NN researcher, since their algorithm fits quite neatly into a classifier role. This demonstrates that the sub-problem of edge-matching is possible, potentially soluble by a general connectionist approach. Furthermore, if edge-matching is possible, then even generalised matching may be practical.

In their concluding arguments, however, Koch et al do state that the analog hardware is very scene-dependent, and has to be re-created for each application it is put to. This would seem to indicate that a general SCP solver would require some ability to redesign its parameters intelligently based upon the input scene. They propose a hybrid architecture to solve this. A schematic of their hybrid massive parallel machine, built using ohmic resistances and transistors to allow some digital processing is shown in Figure 3.15

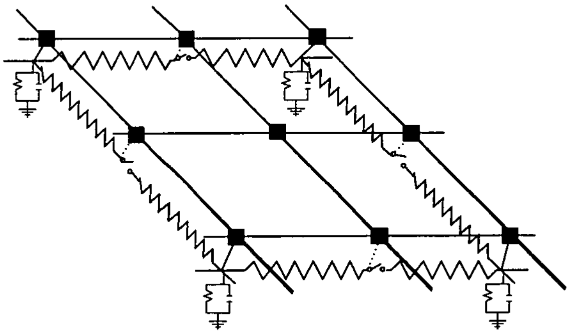


Figure 3.15: *A hybrid massive parallel machine for a real-time artificial vision system.*

3.5.3 Neural Networks and Flow

The last Connectionist solution to be inspected in this section is particularly relevant to this thesis since it concerns the use of Neural Networks to solve Flow Problems.

The paper, by Perfetti in 1994 uses a standard layout Neural Net to calculate the Maximal Flow problem[148]. This problem is at the core of this thesis, the maximum-capacity traversal of a multiply-linked flow network. While not, strictly a vision-based paper, its relevance is even more emphasised by the author's assertion that his NN is suited for VLSI integration, another goal mentioned in the introductory chapters of this Thesis.

Perfetti states that an Artificial Neural Network is convenient if:

- a real-time on-line optimisation is needed,
- the resulting circuit implementation is easy,
- it is easily reconfigurable when the parameters of the underlying problem are modified
- local solutions can be avoided.

The list of characteristics is startlingly similar to the requirements of an SCP solver, as discussed earlier. However, its application to the solution of flow networks only, is Perfetti's aim.

Perfetti points out that classical Linear Programming methods are good at solving flow problems. When one inspects adjacent epipolar lines in a stereo image, one can see that they are very similar, and it is conceivable that a single NN could be used to solve the whole image, simply varying the weights for each line, and recomputing the optimal flow. Perfetti hypothesises that NNs seem very attractive for just such a situation, since they can rapidly adapt to capacity variations or branch changes.

Perfetti shows that the complexity of the Neural Net increases linearly with the number of branches and nodes. A simple approach might be to plot all the disparity matches for each epipolar line in a 2-d grid, such as in Marr's cooperative stereo system, and simulate each point as a node in the NN. This classical approach, however, would require an increase in net complexity, and an increase in relaxation time.

Therefore, extracting a number of smaller "areas of high likelihood", and using the Neural Net technology to reduce these is more efficient. This project does exactly this with the flow networks, and would produce much smaller, and much more efficient NNs.

Perfetti concludes that his method is compatible with analog VLSI implementations, but has the disadvantage of less precision. This, he states is because he uses a simple "penalty" function, to reduce the hardware complexity.

Perfetti cannot claim any neurophysiological validity, since he does not aim at a standard neural problem, and it seems unlikely that this would be a good model for the human stereo system. However the concession to allowing a black-box for purely the flow reduction step seems, at least to this researcher, much more palatable than allowing a NN to solve the whole problem, since the NN is being used as a tool to optimise a single well-understood step. It may be acceptable, therefore to use a NN here, without seeming to side-step the whole SCP issue completely.

3.6 Similar Work

The cooperative stereo algorithm, as outlined by Marr and Poggio is a popular technique for stereo, and many systems have exploited it. For completeness, this chapter will examine several of these similar systems and compare them to this one. It is gratifying to note that other researchers are still attempting to build generic stereo systems using this method as recently as 2001, demonstrating that there is, at least, enough evidence that it is valid to the present day.

3.6.1 A Cooperative Approach to Vision-based Vehicle Detection

Bensrhair et al at the University of Rouen have recently been working on a very similar system to the one outlined in this project[149]. The aim of their system, as described in the title is to assist a driver with the detection of vehicles ahead of a moving vehicle, with a view to taking action if a vehicle becomes too close (again, a similar concept to one of the possible uses outlined in the first chapter). The system is actually a fusion of two different systems. The first is a monocular region-matcher developed by the University of Parma for monocular modelling of vehicles. The second is their own Dynamic-Programming powered stereo matcher system. The system is actually an extension of earlier work by Broggi, for *platooning*, i.e. automatic vehicle control for convoy-following (known as the GOLD system). In this earlier paper the ARGO vehicle is designed, and demonstrated[150].

Bensrhair discusses the problems of keeping the system calibrated[149], and Broggi uses the example of the ARGO system with the GOLD automatic driver to discuss manual and automatic calibration.[150, 151] In their system, they hypothesised two ways to achieve this. The first is useful for general systems such as the one used in this project, whereby the cameras are manually calibrated once. Auto-calibration from that point on is performed by convolving flat textured areas in one scene with a transformation matrix to minimise the differences between it and the same area in the other image. This, Broggi reports, can take as much as five seconds[151], and is unsuitable for a real-time system, especially one where the cameras may be moving around constantly due to vibrations. He noted that the system can "see" a large portion of the car's engine hood, and painted specially marked dots onto it.

The cameras have 360 lines of resolution, and are connected through a high-performance Matrox capture board to a standard Laptop PC running the Linux operating system. The cameras are mounted into a specially designed car called ARGO. Broggi is a co-author of this work. Their requirements are very modest, and report a matching result equivalent to 1.5fps on their hardware.

This result is very impressive, on the order of 20 times faster than the system outlined here, in fact. Since their system is based on similar technology (the runtimes given here are for a 500MHz Sparc system running Solaris), this does seem to make a matcher with a 20 second run-time seem very slow.

It is the preparation step, (before the stereo matching begins) where the massive performance increase becomes clear. The system uses a highly-optimised monocular model-matching step designed by a different group which has been trained to recognise cars from behind. It does this even more rapidly by accepting queues from a "symmetry detector" which works on the premise that cars are by-and-large symmetrical. Once the likely car-candidates have been identified, the images are passed to the stereo vision phase for segmentation and matching.

The results are fairly good, and demonstrate that the system can approximate distance to objects very rapidly. At most distances, the depth error is less than 15%, recovered within a second.

While the core of this system *is* similar, they have tackled a very narrow domain, i.e. car-recognition. The system seems robust and fast, but very limited, due to the many assumptions; flat road, clear conditions, symmetrical regular cars etc.

The most heartening aspect to this paper is its recency and topic. It demonstrates that the network-based co-operation concept, inspired by Marr and Poggio is still a theory being researched, and is generating promising results.

3.6.2 Weighted matchings for dense stereo correspondence

I have selected this work by Fielding since it uses Dynamic Programming, tackles some of the issues which have proven particularly problematic during the course of this project, and seems to give very similar results.

Fielding's paper is actually a comparison of four algorithms: a simplistic pixel-by-pixel "local search", the so-called *left-right* heuristic [2], Dynamic Programming, maximum-weighted-matching, and a "greedy algorithm", which bears a similarity to a graph-traversing shortest-path algorithm.

Fielding attempts to solve the problem of occlusion/parallax confusion which is a particular bug-bear of the system outlined in this thesis. He does so by expanding the *cost of occlusion* method as outlined in Cox's paper "A maximum likelihood stereo algorithm"[64].

He uses all five algorithms against a number of synthetic images, and a natural scene. It is interesting to note that the Dynamic Programming method fails quite dramatically in some synthetic images, where the other algorithms (even the simplistic Local Search) do not. On the natural image, the Dynamic Programming, Maximum-weighted and Greedy Algorithms produce results very similar to what would be output by the system from this thesis. Most interestingly,they all suffer to some degree from the problem of occlusion. By using Cox's cost of occlusion measurement, Fielding has solved some of the occlusion problem, whereby the system mis-detects two occluding objects as a single object which merges with the background (see the boxes dataset in Chapter Six). He does not, however solve the problem whereby a narrowly- occluded area is recovered at the wrong depth. This can be seen between the branches of the tree in his natural image. The background behind the tree (which is a line of trees some distance away) is recovered as being immediately behind the tree itself. This problem also crops up in the boxes dataset later.

It is clear that the "cost of occlusion" works well, and could be incorporated into the algorithm outlined in this project. This will be discussed in the "further work" section.

He does however surmise that in unstructured environments where narrow occluding objects may be present, maximum weighted matching and greedy matching are preferred over dynamic programming.

3.7 Summary

This chapter has discussed a number of stereo vision techniques commonly in use within the field. Their relative merits and problems were discussed, and a number of examples of each of

[2] Whereby the hypotheses gained from using the left image to match the right are used on flipped copies of the images. The hypotheses are used on the flipped-right to try and match back with the flipped-left

the three main types of technique were given.

In conclusion, once again, we can summarise by saying that there is definitely more than one way to solve the SCP, and each way is strongly backed by its own supporters. Many hybrid solutions exist, combining the best parts from each discipline. The performance of the hybrid solutions seems, at least from the results shown in the literature, to produce much improved solutions than the techniques from which they draw their components.

While some may argue that incorporating digital technology into neural net systems removes any neurophysiological validity, others would argue that the aim is to produce a workable, general SCP solution, and validity through recreating biological systems is a "nice-to-have", but not a necessity.

Each of the examples was chosen since it had some bearing on this project. In the following two chapters, the Thesis will investigate the proposed system itself, drawing upon the information given so far.

Chapter 4
Preparing for the Network Analysis

4.1 Introduction

These next two chapters are dedicated to describing the project pipeline itself, which falls neatly into two parts; preparing the data, and performing the Stereo Correspondence. This chapter describes the first section, in which the data is taken from its raw state as read from the camera recording surfaces, to a form whereby the stereo matching processes in the second stage can be executed rapidly and reliably.

Each step is designed to prepare and streamline the data for the following steps. Such a linear design is well-suited to embedding in a VLSI device, and also execution on a general purpose computer. Many of the steps also involve operations on large datasets, which can be easily parallelised.

The data coming from the cameras is typically noisy. For the cameras used in this project, a specific 'intelligent' two step smoothing algorithm was designed to clean the data. The output of this algorithm are two copies of the original stereo pair. These are then used to construct a series of *correlation planes*.

These planes can be used to perform a 'maximum likelihood' stereo correspondence. In order to improve and accelerate the matching, the planes are thresholded and pruned to remove impossible or the least likely matches. These reduced planes are used to construct maximum likelihood paths through the data, which are interpreted to produce the stereo depth maps.

This chapter is dedicated to the first half of the pipeline, i.e. the generation of the correlation planes. Chapter five will describe how the correlation planes are used to generate the stereo depth maps.

4.2 Preparing the data

4.2.1 Smoothing the lines

Many smoothing operators were tested during the course of the project, however the two which provided an optimal CPU usage to results ratio were the Gaussian Smoothing Kernel and the Neighbourhood Averaging Kernel. Smoothing by filtering in frequency space proved far too slow for this application, and was not as easy to implement in hardware or customisable as required.

Recovering depth using pixel intensities requires that the data within the image remain faithful to the original scene. Smoothing operators are sometimes referred to as *filters* since they effectively remove data with certain spatial frequency components. The neighbourhood-averaging kernel, as shown in Figure 4.1 is an example of a low-pass filter.

Running this kernel over the image is equivalent to transforming the image into the frequency domain, and cropping the higher frequencies from the data. With larger kernels, it is often faster to perform this filtering in the frequency domain.

Completely smoothing out all high-frequency information is not desirable, as this removes important data from the images. As Canny found[130], a desirable smoothing kernel should contribute to a pixel value by summing the values of its neighbours, weighted inversely proportionally to their distance from this pixel. Canny calculated the probabilities of pixels having similar values as a function of their distance, and found that a gaussian curve, as shown in figure 4.3 was a suitable approximation.

The matrix in Figure 4.2 shows a 3x3 integer approximation to a two dimensional continuous gaussian kernel. The tail of the gaussian distribution has been removed to produce the 3x3 kernel shown. As can be seen, the pixel in the centre of the kernel is given a much higher weighting than the others. However, the combined weighting of its neighbours means that their contribution is still significant.

The importance of horizontally neighbouring pixels (along the epipolar line) is much greater than that of the vertically neighbouring pixels (on adjacent epipolar lines) in the matching process. Considering this, it seems inappropriate to use a symmetrical smoothing operation upon the images, therefore an asymmetrical smoothing operator was considered.

1	1	1
1	1	1
1	1	1

Figure 4.1: *A simple neighbourhood averaging kernel*

1	2	1
2	4	2
1	2	1

Figure 4.2: *A 3x3 integer Gaussian averaging kernel*

If using a two dimensional smoothing kernel, information from the neighbouring lines is also incorporated into the smoothed result, and therefore contributes to the matching process in the next step. However, when performing the matching along the epipolar lines, information from neighbouring lines is useful mainly for verification purposes only, and should not contribute to the stereo correlation.

For an epipolar line where vertically adjacent lines are dissimilar, such as in the neighbourhood of horizontal edge features, incorporating data from these lines can actually hinder the matching process. Therefore, using an asymmetrical two dimensional smoothing operation is also inappropriate.

Considering these facts, it was decided to separate the smoothing into two stages. First, a highly selective one dimensional discrete gaussian smoothing kernel was applied horizontally to each line, to create the first smoothed image. Second, smoothing was performed vertically using a simple one dimensional neighbourhood average operator, to create the second smoothed image.

Results were variable, and differed only slightly from the standard two dimensional smoothing kernel, with a slight increase in likely matches (15% only). Upon inspection it was found that the excessive blurring of edges due to the necessary width of the gaussian kernel was the remaining point of critical failure as a result of the smoothing process. The gaussian smoothing algorithm was altered such that while the smoothing operation was aggregating results, a differential component was also being computed in parallel. If the differential over a set number of pixels, dependent upon the selectivity of the gaussian kernel, was too high, then the aggregation stopped. This means that smoothing is not carried over edges of a threshold weight and steepness. The result was a slightly more complex smoothing function, which added an extra 18% to the smoothing kernel's runtime, however it improved the quality of the pixel matching operation significantly.

The selectiveness and size of the horizontal gaussian kernel was chosen after much deliberation and study. It is set such that the information loss due to obliteration of the high-frequency components of the data is kept to a minimum, while still achieving a fair smoothing operation.

4.2.2 Creating an average over neighbouring lines

It was known from the outset, that information from neighbouring lines, and indeed, even temporally neighbouring image frames would be an excellent source of correlation information

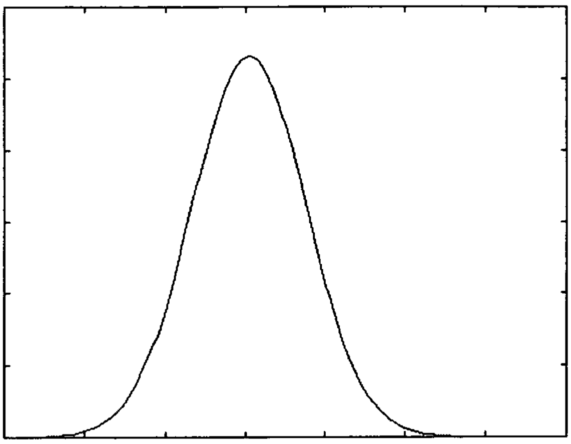


Figure 4.3: *A gaussian curve*

with which to improve the line-level matching.

The process of discovering which matching paradigm was best suited to the task of distilling this correlation information was begun again. Once again, the small noisy components proved to be most troublesome, and it was discovered that information from up to 5 lines either side of the line under inspection was required in order to dampen their effects.

In this particular case, the data from the neighbouring lines is expected to differ from the line under inspection, and the data from this averaging step is used only as a coarse guide to aid the matcher, hence a simple neighbourhood average smoother was more efficient than the gaussian filter while achieving comparable results, and was therefore used.

A smaller kernel is used in this case because the data collected from this pair of images is used purely as an aid to the selection of likely pixels in the next step, and the importance of vertically neighbouring pixels drops off sharply with distance.

4.2.3 Constructing the Correlation Planes

The image data is now in a form whereby robust stereoscopic matching can be performed. In order to do this, a series of two-dimensional *correlation planes* are constructed. Each correlation plane is, in reality, a table of results for a complete comparison between two one dimensional epipolar lines. Since each image contains n epipolar lines, there will be n correlation planes per image pair produced.

Since each input image is smoothed twice, there are two pairs, and hence $2n$ correlation planes of size nxn produced. Figure 4.4 shows this process in more detail.

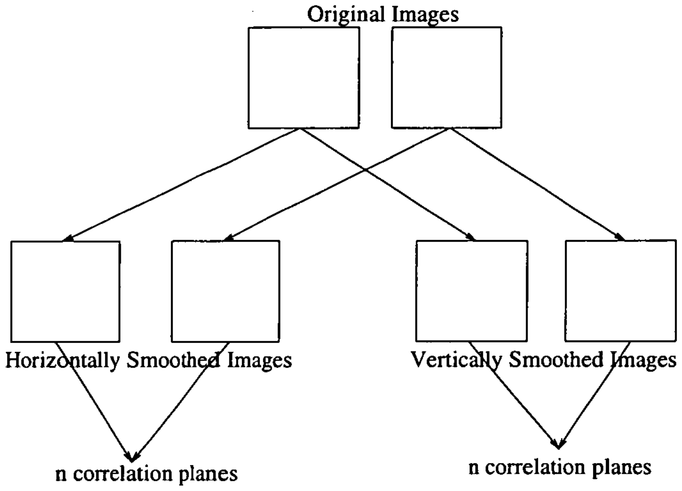


Figure 4.4: *The creation of the 2n correlation planes from the original images*

The generation of each correlation plane occurs in a single, optimised step, combining the two epipolar lines together. However, by deconstructing the process into its component parts, the process can be more easily inspected. Figure 4.5 shows this.

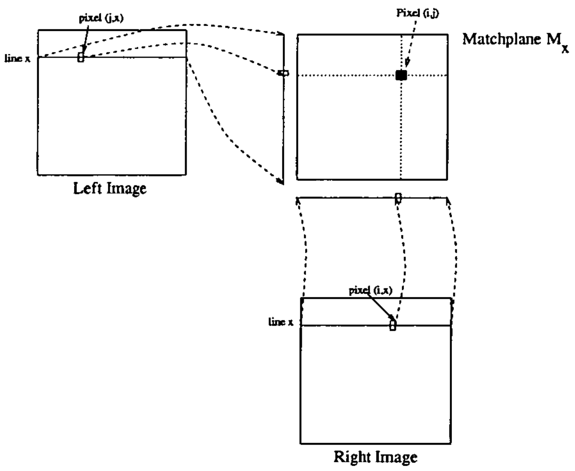


Figure 4.5: *A Graphical representation of the Correlation Plane being generated from a pair of epipolar scanlines*

The first image is constructed by copying the values of the left hand epipolar into the first column of the grid, and then copying the contents of that column right, across the whole width of the grid, achieving the 'smeared' effect that can be seen. The second image is constructed

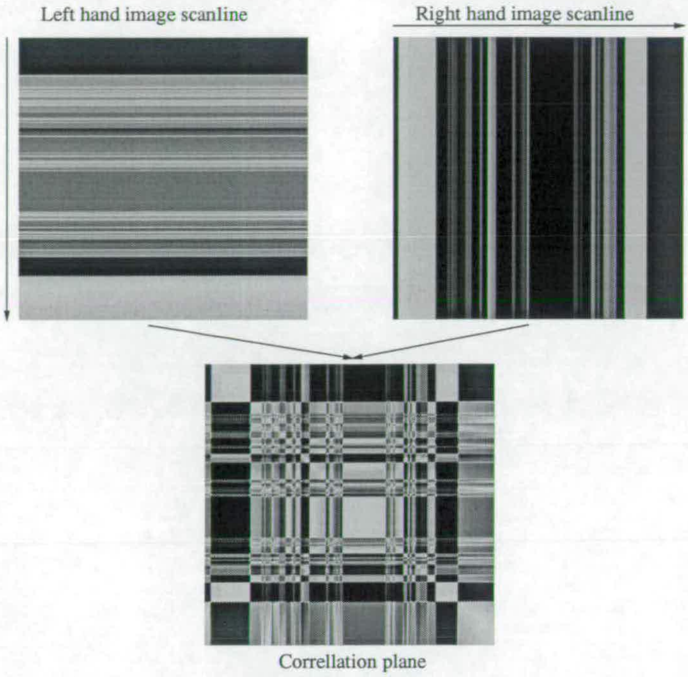


Figure 4.6: *A notional representation of the generation of a correlation plane*

similarly with the values from the corresponding epipolar line in the right hand image copied into the first row along the top of the image, and subsequently copied downwards, achieving the vertically smeared effect.

The resultant plane, is created by performing a simple pixel-by-pixel difference operation between the two planes (M_l and M_r) at each pixel position (i, j), and storing the result back in the resulting grid M_x at pixel (i, j). Figure 4.6 shows this process graphically.

Therefore the following equation describes the construction of the Match-plane which corresponds to the x^{th} epipolar scan line.

$$M_{x(i,j)} = 255 - abs(M_{l(i,j)} - M_{r(i,j)}) \tag{4.1}$$

Since each pixel at position $M_{l(i,j)}$ corresponds to the pixel at position $L_{j,x}$ in the original image, and each pixel at position $M_{r(i,j)}$ corresponds to the pixel at position $R_{i,x}$, then the equation can be rewritten in terms of the original images. Figure 4.5 shows how the match-plane is generated directly from the two original images.

$$M_{x(i,j)} = 255 - abs(L_{j,x} - R_{i,x}) \tag{4.2}$$

If the two pixels under scrutiny are dissimilar, the value of $M_{x(i,j)}$ will be low, close to zero. If the pixels are very similar, the value of $M_{x(i,j)}$ will be close to the maximum of 255.

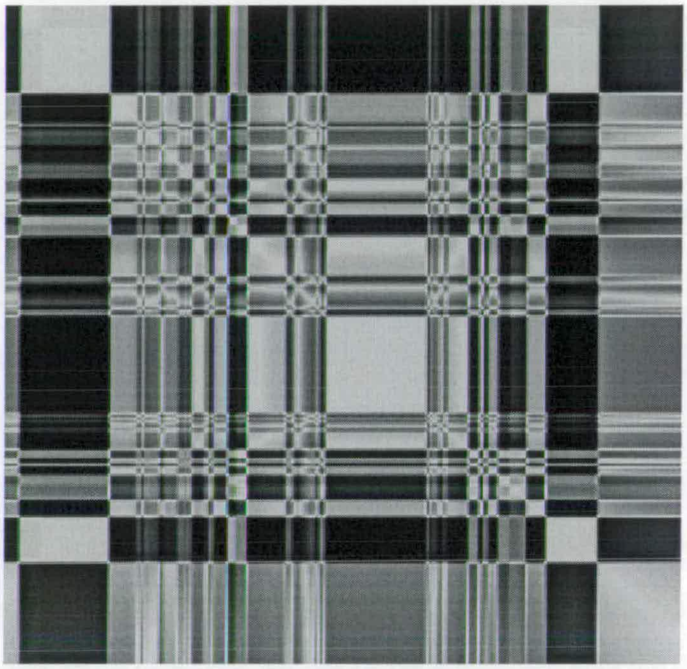


Figure 4.7: *The resultant correlation plane for a single epipolar line*

4.2.4 Thresholding and Pruning the planes

It can be seen from the resultant correlation plane in Figure 4.7, certain areas within these planes have higher pixel intensities, than others. These areas correspond to areas where matches are more likely, since they represent areas where the source pixels have similar intensities, as given by the above equation.

By scrutinising the resultant correlation planes from test data, a set of optimisation-oriented constraints, which can assist the later stages by excising unwanted data, was devised. By imposing these constraints, the number of candidate pixels which must be inspected is reduced to between 1% and 10% of the original.

The first step in this data reduction stage of the pipeline is to threshold the planes. Depending upon how faithfully the cameras can capture the scene, the thresholding level may be between 80 and 98 percent of the maximum match value. For images taken with the DSC-S50 camera, a value of 98% was used. For the images taken with an SLR camera, digital camera or pre-rendered by computer, a value of 99% was used. For images taken with a RS-170 CCD camera, 96% was more appropriate, due to the added noise.

A higher selectiveness will allow fewer candidate matches through the filter, which will speed up the network generation and subsequent relaxation stage. However, if the camera is of poor quality, or the lighting conditions create a poor image, a highly selective filter may remove much data which is distorted just beyond the threshold. It is important, therefore to tune the threshold carefully for the camera used.

In the final implementation of the project, an extra module was built to perform a histogram analysis of the planes as they are built. By analysing the distribution of the qualities of the matches, it is possible to specify exactly what percentage of the data should remain. It was found that with this extra module, delivering a dynamic value to the thresholder, based upon the individual image characteristics, the system had a better pruning response to images of all intensities, giving the system a wider response to real world environments. Figure 4.8 shows the histogram/selection/thresholding process pipeline.

The thresholding operations given above are performed on both sets of smoothed images. However, the pruning, and block removal sections which are to follow are performed only upon the horizontally smoothed (primary) planes. The vertically smoothed planes are brought in again after the block removal, to assist in the final removal of unwanted match candidates. The algorithm was first designed to perform pruning and block removal on both sets of smoothed planes, however it was found that equally valid and much more rapid results were obtainable without continuing the data reduction exercises any further on the vertically smoothed images.

The next step is to prune away the match candidates which are either impossible, or simply highly unlikely. By inspecting the geometry of the camera setup, as shown in Figure 4.9 it can be seen that 50% of the data can be removed immediately.

Consider an object A in the world. When projected onto the two viewing planes A will produce the images I_{LA} and I_{RA} in the left and right images, respectively.

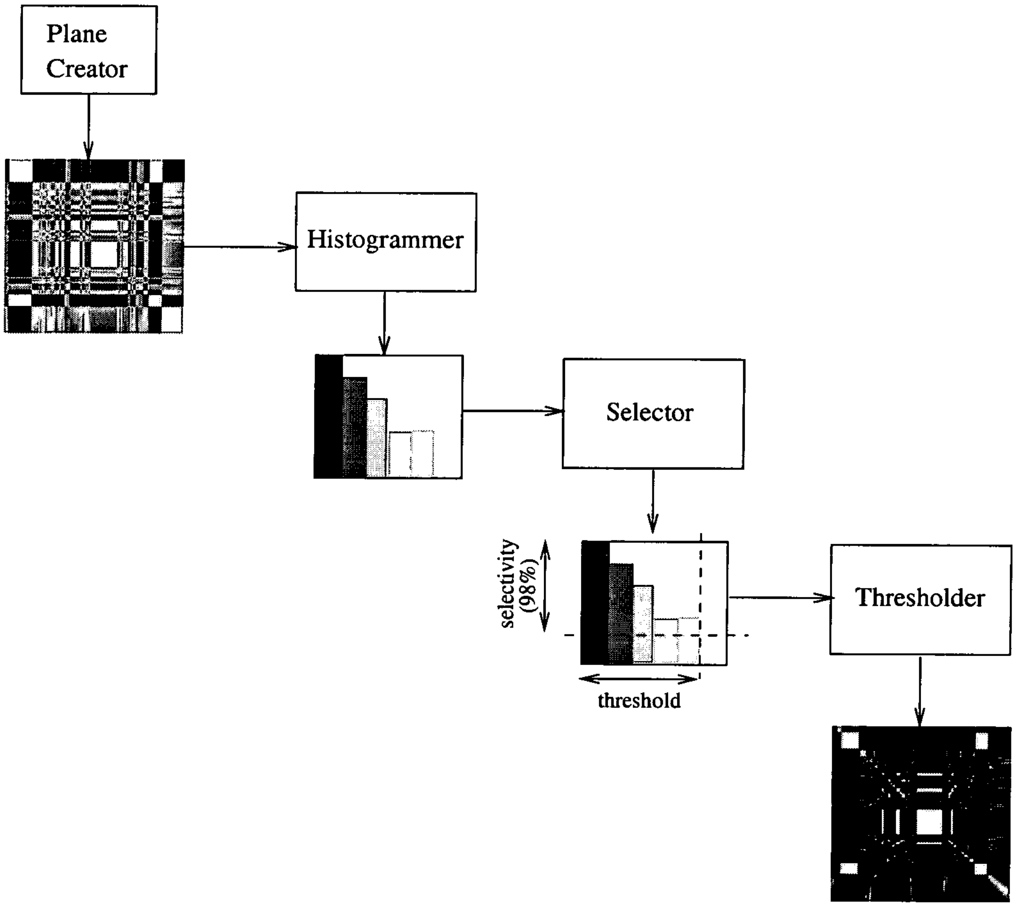


Figure 4.8: *The process by which a dynamic thresholding is achieved, using a dynamic threshold selector*

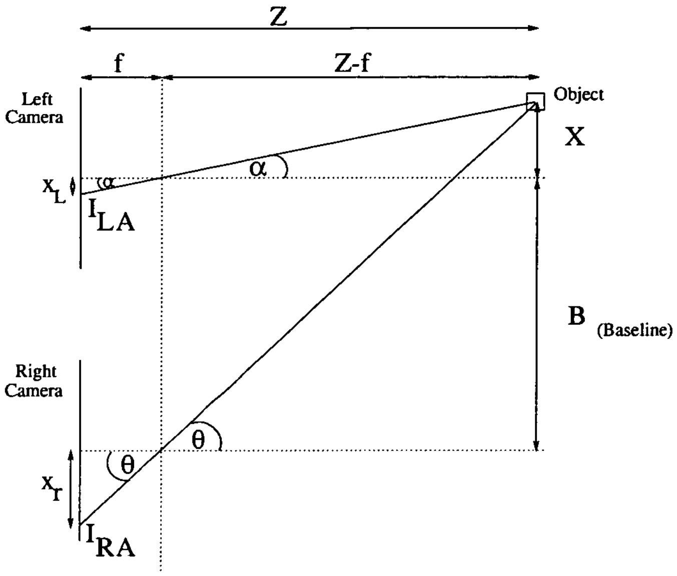


Figure 4.9: *Geometry of the stereo rig*

The object is at coordinates (X,Y,Z). The image I_{LA} appears at location (x_L, y_L). The image I_{RA} appears at location (x_R, y_R). It can be seen that in the left hand image,

$$tan(\alpha) = \frac{X}{Z - f} \tag{4.3}$$

and that

$$tan(\alpha) = \frac{x_L}{f} \tag{4.4}$$

therefore

$$x_L = \frac{f.X}{Z - f} \tag{4.5}$$

One can also calculate the coordinates of x_R in a similar fashion.

$$x_R = \frac{f.(B + X)}{Z - f} \tag{4.6}$$

since B,f and Z are always non-negative, it is safe to say that

$$\frac{f.(B+X)}{Z-f} > \frac{f.X}{Z-f} \tag{4.7}$$

Because the lens causes lateral inversion, when the data is sent to the computer, the camera performs a horizontal 'flip', to correct the effect. Once the images are stored in an internal format, the origin of the axes is moves to the top left of the image. This is to accelerate the raster image processing. Figure 4.10 shows the left and right image views of an object, which has been recovered. As can be seen, once converted into the top-left centred coordinate frame, the object 'A' has a greater X value in the left image than in the right.

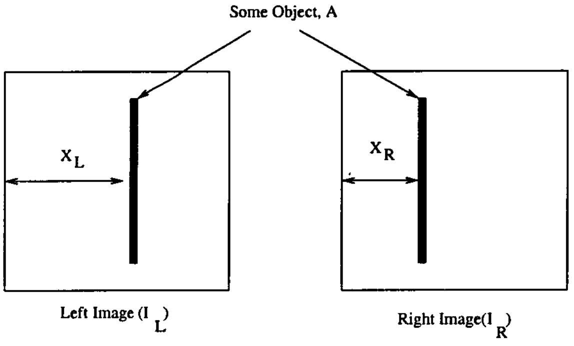


Figure 4.10: *The disparity of a recovered object*

This means that no matter where the object lies in space, as long as it is within the view of the cameras, objects in the left hand image will always have a greater x coordinate than the projection of the same object in the right hand camera.

This is a side-effect of having co-planar cameras. If it is expected that the cameras will not be coplanar, then this optimisation must be removed, since it is no longer possible to assume that all matches will have greater x values in the left image than the right. The approach can still be applied but will be significantly slower since the network generation step now has to consider more matches.

The match planes represent *all* matches, including those between objects whose projections would be such that their images appear with a greater x coordinate in the right hand image. These matches are only possible if the object is further away than infinity, which is impossible, or behind the cameras, which is similarly impossible, since an object behind the cameras cannot project an image onto its recording surface.

Therefore all comparisons for impossible matches are removed from the correlation plane. Figure 4.11 shows the plane after the impossible matches have been reduced to zero.

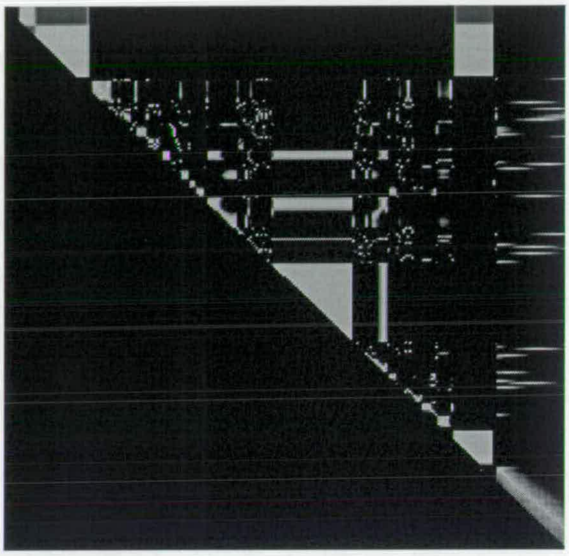


Figure 4.11: *Thresholded match plane with impossible matches removed.*

Once the impossible matches have been removed, it now remains to remove the *unlikely* matches. These matches fall into two categories.

Firstly, there are matches which correspond to objects very close to the cameras. One of the main assumptions that vision researchers make is that objects will be presented to the camera only within its range of focus. By studying the images formed by objects at varying ranges, it was found that objects closer than 10 centimetres became blurred, but objects closer than 25 centimetres could not be detected in both images if the cameras are arranged with a separation of 50 centimetres, this distance corresponded with a stereo disparity of roughly 50% of the width of the image. Therefore if one places the constraint that the camera will not approach closer than 25 centimetres to any object, a further 50% of the data can be cleaved. Figure 4.12 shows the plane after the unlikely matches have been removed. These values of course change with the camera geometry and properties.

Secondly, as can be seen from the image in Figure 4.12 there are many single or small groups of bright pixels, indicating matches, scattered over correlation plane. Very small groups of pixels

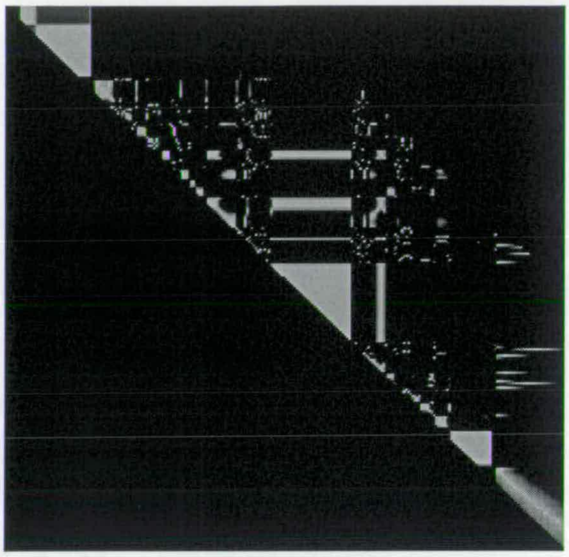


Figure 4.12: *Thresholded match plane with impossible and minimum distance constrained matches removed.*

are highly unlikely to represent the correct match, especially if they are contending with a larger block. Also, the *coherence* constraint binds the solution of the SCP to objects whose images form coherent blocks. Therefore single pixels, or small disparate groups are equally unlikely to form the basis of a correct match. Therefore, these outlier blocks below a certain threshold size (3x3 or 5x5), which are not within a certain distance from another block are stripped from the data.

Figure 4.13 shows a grid of pixels. A simple 5x5 kernel is therefore passed over the non-zero pixels within the grid. The grey levels for each of the non-zero pixels within the bounds of the kernel are summed, and averaged. Figure 4.14 shows the result of this averaging operation.

Based upon a restriction that a pixel is valid only if there are three other pixels of at least value 225 within its 5x5 neighbourhood, this means that only pixels whose values are greater than 38 after the kernel has passed are valid. As can be seen, the pixel clusters in the top left and top right of the grid are too small, and are therefore removed. The slightly larger cluster to the lower centre of the grid is large enough to be a possible area for further scrutiny and therefore survives.

Figure 4.15 shows the matching plane after this step. As can be seen, the remaining matches are a small fraction of the original plane.

0	0	0	0	0	0	0	0
0	250	0	0	0	240	245	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	240	240	0	0	0
0	0	0	240	240	240	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Figure 4.13: *an example of one area of the correlation plane*

0	0	0	0	0	0	0	0
0	10	0	0	0	20	20	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	48	48	0	0	0
0	0	0	48	48	48	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Figure 4.14: *An area of the plane after the non-zero pixels have been averaged.*

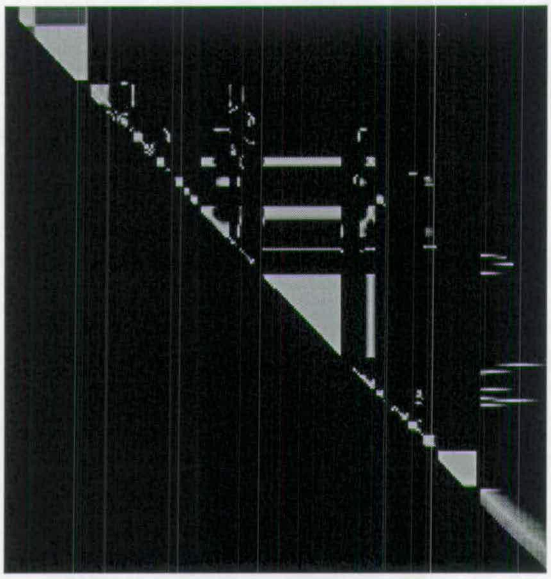


Figure 4.15: *Thresholded match plane with impossible, improbable and minimum distance constrained matches removed.*

Performing this pruning operation takes time. As will be shown, this cost is recouped adequately later, in the network generation stage.

There is an initial start-up cost for each network created in the next step, due to the allocation of memory and the reordering of the representation such that the network is added to the search space. Each individual group of pixels creates its own network, and therefore the removal of the smaller groups, and the less likely groups at this stage saves time later.

4.2.5 Block Removal

By inspecting figure 4.7, one can see large blocks of pixels which have been detected as having a high probability of being a match candidate. However, the Uniqueness constraint tells us that for each pixel in the left image there should be no more than one match in the right image. This means that most of the pixels in the large blocks are unlikely matches.

A *Block Detector* is run across the image, and the co-ordinates of all blocks, larger than 9x9 pixels are stored in a queue. These co-ordinates are used to inspect each of the blocks closely. If they are largely featureless, then this implies that they represent a match between two featureless areas in the images.

When this occurs, the most likely paths are clustered around the diagonal. Unfortunately, one side-effect of occlusion, parallax, and the decimation steps previously is that the blocks may have outlying corner pixels pruned. Therefore, it is not possible to simply reduce the blocks to their diagonal. By examination of test data, it can be seen that larger blocks are often perturbed by previous steps more than small blocks. This is true probabilistically, since a larger block is more likely to be the victim of an occlusion, and more likely to have some outlying 'trailing' pixels which are pruned.

Therefore each large featureless block is removed, and replaced with a diagonal line of pixels whose thickness is proportional to the original block size. Figure 4.16 shows the result of removing the diagonals of a group of blocks.

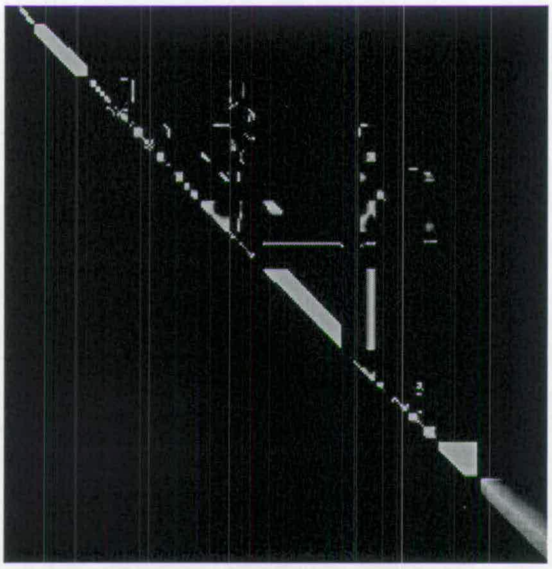


Figure 4.16: *Match plane after final stage of unlikely match removal. The large blocks of data have been reduced to diagonals only.*

If a diagonal thicker than a single pixel is used to replace the block, then, when the algorithm chooses the best path through the entire network in the next section, it can select which path (along, or close to the diagonal) most suits the matching in that context, maintaining the maximum likelihood probability of finding a good match. Therefore, it restricts the number of paths to be considered, but still leaves some room for uncertainty.

Of course there are going to be some oddly shaped blocks, and blocks which merge into each other. Much study was done with real and generated images to detect what happens in just such occasions.

The most common occurrence is when there are a few extra pixels at the edges of the block, giving it a ragged edge. This is caused by quantisation errors, noise or simply an unfortunate occlusion between two similarly coloured objects. It was determined that in this case, the best thing to do is to trim the block first, and then remove the block as before, making the diagonal replacement slightly larger than normal for the block size.

4.2.6 Hysteresis: The use of neighbouring line information

For most images, the horizontally smoothed planes are now almost completely reduced to a single possible path. However, it is still possible that there is some ambiguity, and it would also be useful to allow the neighbouring lines to assist in the choosing of the paths, for continuity. Therefore, it is at this stage, that a hysteresis step is used.

A match is kept in the horizontally smoothed images, only if there is also a match of high intensity in the vertically smoothed match plane. Since the horizontally smoothed images have had far more data reduction steps applied to them, only a small amount of data is removed at this stage, however it is a fast raster operation, and generally results in fewer outlying single-pixel matches.

The removal of information by this hysteresis method tends to break up larger blocks, as well as obliterating smaller ones. This is undesirable if one is subsequently to attempt to remove blocks, since they will be that much harder to detect. For this reason, the hysteresis step is the last in the data removal pipeline.

4.3 State of the Data

At this point, there are 256 grey-level planes, each 256x256 pixels in size, One plane for each row in the original pair of images. The planes have undergone a rigorous reduction process, designed to operate at a fair speed, yet still maintain the maximum likelihood properties.

If the system were to be presented with a perfect image, to a perfect pair of cameras, the next stage would be to simply read off the pixels remaining in the planes in order to obtain the depth information. However, despite the best efforts of the smoothing, pruning, thresholding and hysteresis filters, there will still be some ambiguity, and some erroneous data due to the imperfect nature of the world scene presented. In order to best deal with this, the system employs networks, most specifically, flow networks.

Flow networks have the advantage of allowing a user to divine a minimal-cost path through a complex 'environment'. They are highly flexible, robust, and comparatively easy to prototype for use on a general purpose machine. As will be seen in the next chapter, they are also a very attractive method here because they can be joined together very easily, and can be re-programmed with different, and even seemingly conflicting priorities.

Chapter 5
From Networks to a Depth Map

5.1 Introduction

In Chapter four, the process of cleaning, reducing and preparing the image data was described. The result of those processes is a set of highly optimised 'match planes'. Each match plane is a two dimensional grid of values representing the set of all possible correlations between two epipolar scan lines. Since the scan lines are collinear with the raster scan lines of the images, which are 256x256 pixels, there are 256 match planes, each representing the correlation between two 256 element arrays, and are therefore 256x256 elements in size. This can be seen more clearly in figure 4.5

In this chapter, these match planes are studied using network algorithms in order to reduce the set of possible matches down to a small set of probable paths, and then to construct a super-network in order to find the optimal route through these probable paths. This yields a single maximum-likelihood path, which can be 'read off' as a stereo disparity, and which, using trigonometry, can be converted into a 'depth map', as described in Chapter Two.

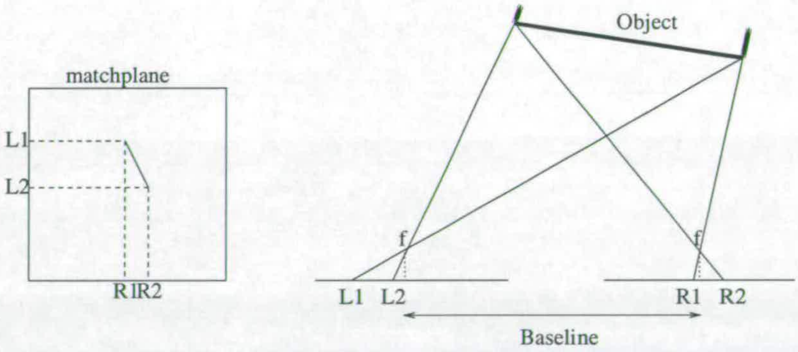


Figure 5.1: *reconstructing the 3d position information by reading off the correlation position within the match plane using the known camera parameters and trigonometry*

5.1.1 Building the Networks

In this application, each match plane from the set of 256 is taken individually.

Each plane is inspected, and for each possible correlation, as denoted by a high-valued cell (shown as bright pixels in the planes), a vertex is created. Each vertex is created with two properties, a location within the plane, and a weighting, which is proportional to the value of the cell which it represents.

Once the initial set of vertices is created, the set is ordered with respect to 2-D position within the plane using Quicksort. Once ordered, each vertex within the set is inspected, and its co-ordinate position compared against the other vertices within the set. If the two vertices are closer together than some threshold distance, an edge is created between them, and given a weighting. Figure 5.2 shows this process.

There are certain "zones" within which links are allowed to form. This is to enforce the continuity constraint, and is shown in Figure 5.1.1. It is for this reason that networks 1 and 2 cannot join together. There are no nodes in network 2 which are within the allowed zones for any of the nodes in network 1. The same applies when attempting to connect network 2 to network 3, and network 3 to network 4. While network 4 is within the allowed zone for networks 1 and 2, it is too far away to form a viable connection. Therefore networks 1, 2 and 4 remain independent.

Algorithmic runtime is closely related to the sizes of the networks created. Therefore an upper limit of 400 nodes is placed on the networks. If a network is created larger than this, each node in the set is passed through a "selector", if the node is worse than some threshold value (calculated as a percentage of the maximum node value), the node is cleaved from the network. If the network is still too large, the network is run through the selector again with the selectiveness increased by 1%. The selector can be configured to begin with any value. Since the nodes are initially chosen from the correlation plane with a selectiveness of 97%, the selector will begin with a value 1% higher than this. The selector is repeatedly run until the selectiveness is 100%, or the network drops below 400 nodes. Performing this data reduction decreased the run-time, while marginally reducing quality of results. If the network is cleaved into discrete sections by this process, smaller networks are created in its place.

The edge weightings are calculated by inspecting each of the pixels through which the edge would pass, and averaging the values along the path into a single value.

Figure 5.3 shows a typical edge which notionally passes through a number of cells. In this case, the edge passes through 6 cells, including the start and end cells, therefore its weighting is a weighted average of the values of these cells. The average is weighted because the line does

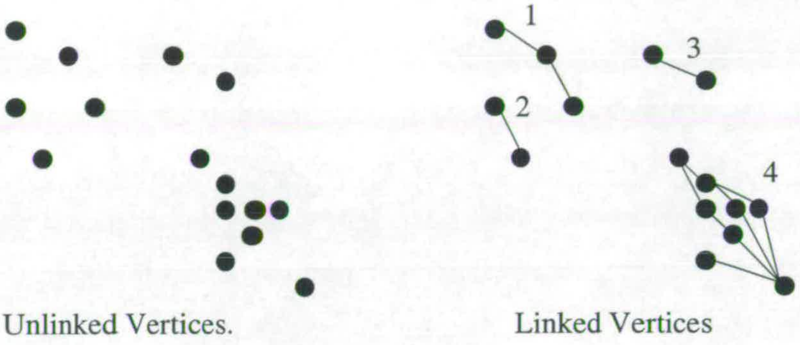
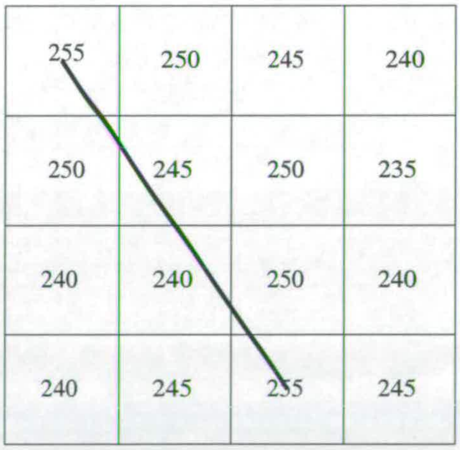


Figure 5.2: *Linking the vertices together with edges*



255	250	245	240
250	245	250	235
240	240	250	240
240	245	255	245

Figure 5.3: *An example edge*

not pass directly between or through the cells. The intersection value of the line with each cell is calculated, and this value multiplied by the cell's value. Figure 5.4 shows an edge passing between two cells. The line's path takes it further through the left hand cell than the right hand cell. In this particular case, the left hand cell contributes triple to the weighting average, than that which the right hand cell contributes. Therefore the weighting contribution from this pair of cells is $(0.75 * 230) + (0.25 * 250) = 235$. This process is known as *anti-aliasing* in computer graphics. It is typically used to remove the *jagged* appearance of lines when they are drawn on raster displays. This technique is not normally coupled with flow networks. It is used here to give faithful values for the edge weightings, and possibly allow some sub-pixel accuracy to assist in the matching.

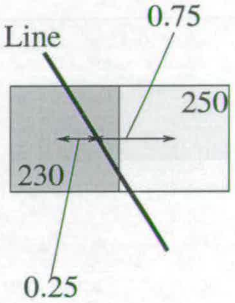


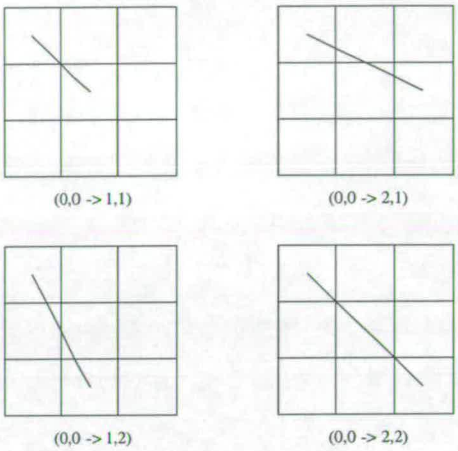
Figure 5.4: *Calculating the Weighting of an edge*

Calculating all of the intersection values takes valuable time. Once it is realised that there will be only ever a finite number of edge types due to the various constraints, it can be seen that these intersection values can be pre-calculated. The system operator can set the maximum length that an edge can have.

By experimentation, it has been found that the furthest jump which allows for a rigorous decimation step (as described in Chapter Four), while maintaining the coherence constraint is 3 pixels. This value, is in part based upon the outlier removal stage.

Therefore to link a match candidate to another, only those within 3x3 cells in the lower right (positive x, and positive y) quadrant are considered for edge linking. This can be increased to 4x4 or 5x5 in cases where the camera produced extremely poor results. Doing so, however, adversely affects runtime.

Since horizontal or vertical matches indicate a violation of the uniqueness constraint, this means that there are four possible edge types, for a 3x3 maximum jump, as shown in figure 5.5. This means that it is trivial to precompute the weightings, and saves much time later.


(0,0 -> 1,1) (0,0 -> 2,1)

(0,0 -> 1,2) (0,0 -> 2,2)

Figure 5.5: *The four possible edge types*

From this point on, since edges are being constructed between match candidate cells, it is beneficial to think of those match candidate cells as *vertices*. Since each vertex can be linked to from previous vertices, and can also, itself link to others, each vertex is therefore said to have two sets of links, a *source* set and *sink* set. Figure 5.6 shows a typical vertex after it has been connected to its peers.

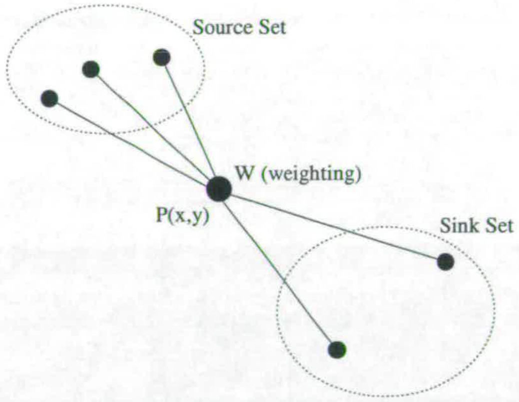


Figure 5.6: *A typical Vertex*

The *source* links connect the vertex to others with lower coordinate values. Since the coordinate system has its origin in the top left of the match planes shown here, this implies that Source vertices correspond to matches with lower x coordinate values on the original epipolar lines, and are within the upper left quadrant, as shown in Figure 5.1.1. The *sink* links connect the vertex to others with higher coordinate values, i.e. matches with greater x coordinate values on the original epipolar lines. These are found in the bottom right quadrant, as shown in 5.1.1.

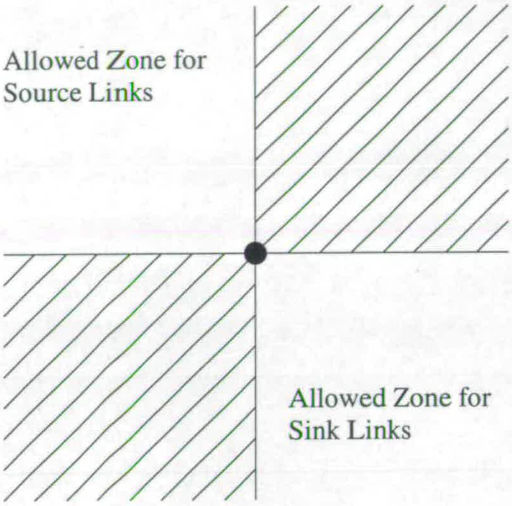


Figure 5.7: *The zones within which Vertices are allowed to form links*

Because the vertices can only link to other vertices which have a greater x and y coordinate within the plane, cyclic networks are impossible, enforcing the uniqueness constraint. Also, because of the way the epipolar lines are used to construct the match plane, this enforces correct ordering as well. This by-product of the method saves much time and computational power over systems which must explicitly check for these two constraints.

When the vertices are connected together with edges, groups of networks are typically formed. Rarely does one single large network form, as this occurs only in situations of perfect lighting, perfect cameras, and with a very simple scene.

Without the previous reduction steps, each network would represent *all* possible matches for a group of pixels formed from an object at a certain depth. However, since the planes have been passed through the data reduction steps, given in Chapter 4, each network represents only the *most likely* candidate matches for objects, or segments thereof.

One unfortunate side effect of presenting the system with natural scenes, is that the threshold for each data reduction step cannot be set exactly, only approximately. The system does employ a feedback mechanism such that the thresholds are set by the outputs from the previous steps, as described in Chapter Four, but this is still uncertain. The distribution of the desired data within the plane demonstrates a coagulation into *blobs*.

These blobs are due to the coherence of objects. If the image of an object, projected onto the recording surface of a camera, is a certain colour at a certain point, then it is a high likelihood that many of the pixels in its vicinity will be similar intensities. If one thinks back to the generation of the match planes, a group of similar-intensity pixels in one image, when matched with another group of similar-intensity pixels in the other image will produce a block-shaped area of similar intensity. After the information decimation steps, one is left with blobs of high intensity.

5.1.2 The Raw Networks

Since the data typically occurs in blobs, the result of the edge linking is a group of networks. Most of these networks consist of highly-connected, densely packed vertices. Other, more accurate matches, or smaller objects, produce less highly-connected networks.

If the cameras are good, and the data near-optimal, there will only be a small number of candidate networks. Those networks will consist of long paths from their source vertex to their sink vertex, will be moderately linear in appearance, and there will only be a small number of valid paths, possibly as low as one, for each network. Also, there will be only minor gaps between the sink vertex of one network, and the source of the next.

If the cameras are poor, or the data noisy or complex, the networks will be square or circular in appearance with many short paths through them. The networks will have many possible sources and sinks. They will also be small, and separated by long distances from others.

It is hoped that the pre-processing steps will have reduced the number of poor quality networks. For example, the block reduction step removes all rectangular-aspect poor networks and replaces them with their diagonals. This is the equivalent of taking a poor network at this stage, and stripping off the most unlikely paths to convert it into a more suitable network.

Now that the networks have been created, focus can turn from the raw data to solving a well-known mathematical problem, finding the optimum path through a weighted directed acyclic graph (or flow network).

5.2 Using the Networks to find the Optimum Path

5.2.1 Reducing the Networks

The emphasis thus far in the processing pipeline has been one of identifying the most likely pertinent data, and removing the least likely. At this point, there is typically less than 5% of the original data left under consideration. However, the final solution is approximately 0.4% of the original data (given images of 256x256 pixels). Therefore there is still more than a ten-fold reduction necessary.

Each path through each of the networks generated in the previous section represents one *possibility*. Depending upon the underlying pixel values, surrounding matches, and the other networks to which a certain path would 'attach' in a final solution, certain paths are considered better than others.

Dykstra's Algorithm, described later, which is used to reduce the networks to a single lowest-cost path is elegant and simple. Unfortunately, it relies upon the assumption that the edge weightings remain the same while the reduction process continues. The main problem here is that the best path through a network often depends upon the networks around it. Therefore, before one can begin the reduction a further step is required.

5.2.2 Network Contention

Often in images, a repetitive texture, or several similar objects will cause multiple matches to appear to be possible for a given epipolar line. When the match planes are produced, and finally the networks generated, these alternatives appear as other networks which appear to be *in contention*. Since the uniqueness constraint says that for any pixel there can be only one match in the other image, two networks which are in contention must be compared for validity, and the loser removed from the network pool.

Figure 5.8 shows three networks in contention. The deciding factors in which network is the most likely are:

- Average edge weighting
- Average vertex value

- Proximity to other networks
- Size
- 'detour' value.

The average edge weighting and vertex value are calculated while the network is being generated, and are free by-products of the linking stages. The proximity to other networks is calculated from the source and sink vertices. Each of the networks in the *pool*, is stored with a bounding box, calculated from the source and sink vertices. This is then used to sort them and calculate their size factor.

The *detour value*, is a notional variable which is calculated from the effect that choosing that network would have on the rest of the result. For example, by choosing network 3 in the contention example, the two networks marked "A" and "B", would have to be skipped completely because of the smoothness constraint. This would lead to a large "hole" in the resulting depth map. When each of the networks are considered, it is discovered whether there would be any valid data that could fill in the "hole", i.e. networks A and B, and if so, it is given a detour value detrimental to its likelihood of any future consideration. This detour value is relatively small, and used only in cases of a tie between two candidates, since it takes some time to search the network pool for candidates.

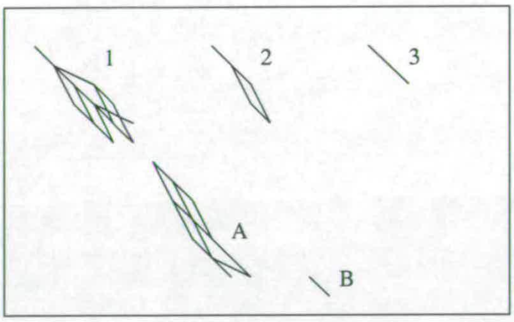


Figure 5.8: *Three networks (1,2 and 3) in contention. Two other networks (A and B) are used for context.*

During this step, any networks of two vertices or smaller (which have slipped through the decimation steps) are removed completely. Also, any networks of size 3 or smaller, which are in contention with a network of size 15 or larger are immediately discounted. It is debatable whether this is a beneficial step, but it provides some performance boost to the sorting and contention subroutines.

Once the smaller, outlying networks have been removed from the contention, the larger networks are then processed for their minimum cost paths.

5.2.3 Obtaining the Minimum Cost Paths

Many algorithms exist which can find the *minimum cost path* through an acyclic network. One of the most respected, and most studied is that of **Dykstra's Algorithm**. Dijkstra's algorithm solves the single-source shortest-paths problem on a weighted, directed graph $G = (V, E)$ for the case in which all edge weights are nonnegative. Therefore $w(u, v) \geq 0$ for each edge $(u, v) \subset E$ [134].

Cormen et al describes it thus: Dijkstra's algorithm maintains a set S of vertices whose final shortest-path weights from the source s have already been determined. This is, for all vertices $v \subset S$, we have $d[v] = \delta(s, v)$. The algorithm repeatedly selects the vertex $u \subset V - S$ with the minimum shortest-path estimate, inserts u into S, and relaxes all edges leaving u [134].

In the following implementation, a priority queue Q is maintained that contains all the vertices in V-S, keyed by their d values.

```
$DIJKSTRA(G, w, s)$
    $INITIALIZE - SINGLE - SOURCE(G, s)$
    $S \leftarrow 0$
    $Q \leftarrow V[G]$
    while $Q \neq 0$
        do $u \leftarrow EXTRACT - MIN(Q)$
            $S \leftarrow S \cup u$
            for each vertex $v \subset Adj[u]$
            do $RELAX(u, v, w)$
```

Obtaining the optimum path is not a simple matter, however. If it was simply a case of following the highest intensity edges through the network, then a simple sort and select would have yielded a result from the initial data in a fraction of the time.

In some cases, where camera noise, or a small vertical occlusion, fouls the data, there can be a small section of low-quality matches within a network. It is important that any optimum-path-finder understand that these areas exist, and is able to calculate whether their impact outweighs the benefit of the rest of the path.

It is also important to understand that Dykstra's Algorithm works on networks which have a single source and single sink. As can be seen from the above examples, networks often have multiple sources and sinks. In order to overcome this, a super-source and super-sink vertex is created, and the various sources and sinks linked with weighted virtual edges to their super source or sink.

The weight for the virtual link is a simple value derived from the source or sink's position. The higher x and y value any sink has, or the lower x and y value a source has, the greater is the link value. While this is not optimal, efforts to base the virtual link value upon *suitability* were unsuccessful, since the suitability would be based upon inter-network link values derived from the super-network stage, which comes later. Because Dykstra's Algorithm requires the vertices to have static, and pre-determined values and edge weights, the super-network can only be performed *after* the networks themselves have been reduced, to find the vertex values for that stage. *Ergo*, one has a cyclic dependency problem.

Once the networks have stabilised, and the optimum paths discovered, they are given a final quality value based upon the quality of the path, and the amount of image that the network spans.

$$NetworkExtent = \sqrt{\frac{(SinkY - SourceY)^2}{(SinkX - SourceX)^2}} \tag{5.1}$$

$$Quality = \frac{AverageLinkWeight}{NetworkExtent} \tag{5.2}$$

5.2.4 Creating the Super-network

The so called *Super Network*, is simply a network whose vertex values are the quality values given to the networks earlier. Each of the vertices in the super network is joined together with edges. Each of the edges denote that a connection is *possible*. A link between two networks, say A and B, is determined to be possible when the x and y values of network A's sink are both lower than the x and y values of network B's source vertex. The weight of the link is determined by the distance between the source and sink vertices.

Figure 5.9a shows a typical group of networks, as could be produced by a stereo system. Figure 5.9b shows a super-network which could be produced from these. As can be seen, each of the sub-networks are reduced to just 'node' status, although in actuality, the nodes are soft 'links' to the original networks themselves, so that path traversal can be performed on the super-network transparently.

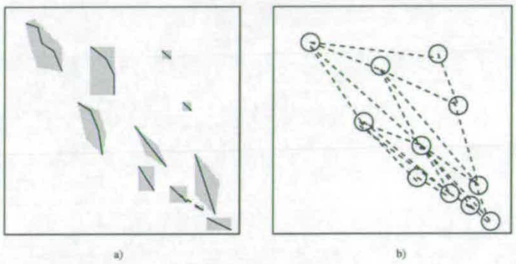


Figure 5.9: *Super Networks. a) Networks, as output by the initial stages of the system. b) The super-network formed from them.*

The super-network's super-source is given the co-ordinates (0,0). It's super-sink is given the co-ordinates (n,n) (where n is the width of the source images, i.e. 256 in most test cases). If there are any networks bordering on the edge of the match plane, then the source and sink are moved to match their source and sink respectively. This improves accuracy at the edges. When multiple networks border the edge of the space, i.e. in a contention situation, an average position is taken between them to keep the selection fairly based upon the quality of the networks, and not just upon their position at the edge. Doing this was found to improve the source and sink anchoring at the edges of the image.

5.2.5 Reducing the Super-network

All flow network reduction is produced within the system using a generalised flow network reduction engine. Therefore, the super-network is created within computer memory using exactly the same programming components as the smaller networks earlier. This abstraction simplifies coding, and makes the program much smaller.

Since the super-network may be as complex as a smaller network, depending upon image complexity, little overhead is wasted.

Therefore, the super-network, once created is passed through the reduction engine, and the resulting network is an optimal super- network.

5.2.6 Depth Map Production

The final stage in the pipeline is to generate the actual depth information itself from the networks. Held within each node of the hierarchical conglomeration of vertices is positional information from the match plane. The co-ordinates of the vertices themselves, when put through a trivial subtraction operation give the displacement of the image.

5.2.6.1 Inter-network spaces

The operational data storage components within the system are highly hierarchical. Once the super network has finally been reduced, it is then necessary to traverse down the tree, unpicking the chosen paths from the hierarchy, and using them to recreate the Depth Map.

The first step, is to use the super-Network's inter-node links to fill in the gaps between the sub-networks. If one thinks back to the decimation step, it is possible that networks may become isolated from each other. In this case, it is necessary for the system to *fill in* the intervening data, where required. This is done using a simple line-drawing algorithm.

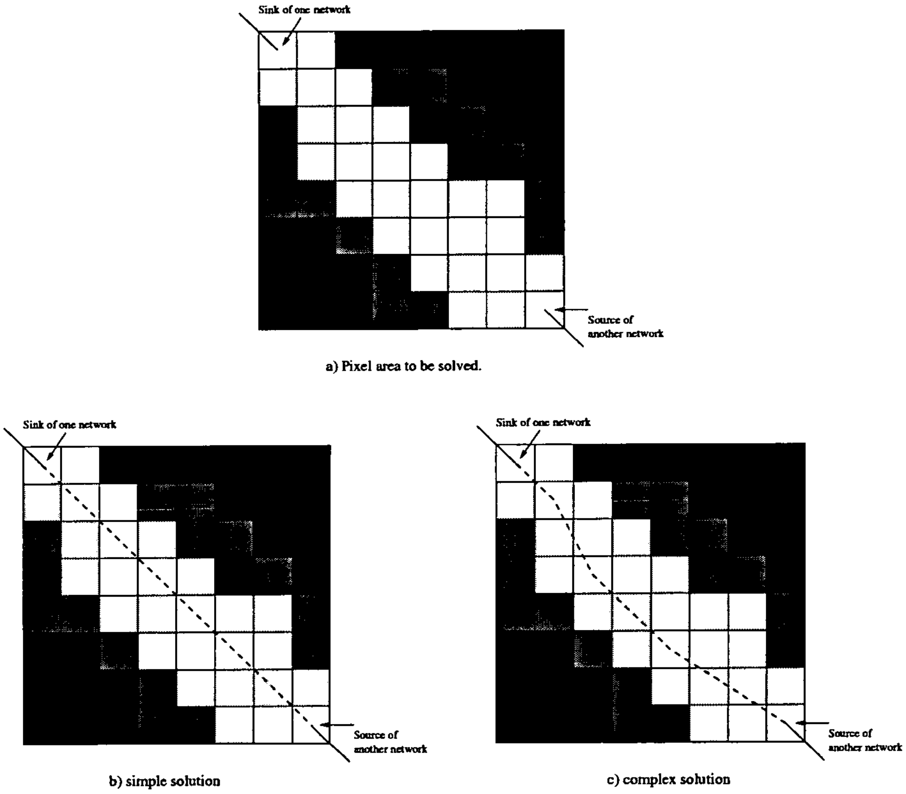
It begins at the sink node of a network, and proceeds towards the source node of the next network in the super-network. Two methods have been tested, and the second has proven to give superior results. Figure 5.10a shows the possible layout of pixels between two networks. Darker pixels indicate poorer matches.

The first method is to average the link strength across the gap, and if it is above some threshold level, the link is granted, and the intervening pixels marked as matches. This is demonstrated in figure 5.10b.

The second method is to check the two pixels either side of the 'line', and choose the better of the two, if it is above a threshold value. This is shown in 5.10c.

As each pixel is "chosen", or as each pixel is interpolated over, its disparity is calculated from its position, and the resulting value placed into the disparity map.

If the first method is chosen (what this researcher thinks of as the all-or-nothing method), if the


Sink of one network
Source of another network
Sink of one network
Source of another network
Sink of one network
Source of another network

a) Pixel area to be solved.

b) simple solution

c) complex solution

Figure 5.10: *A possible topology of pixels between two chosen pixels. b) Simple solution c) Complex solution which inspects pixel values.*

pixels are of insufficient quality to be considered matchable, then a special token is placed in the disparity map, to signify no match.

5.2.6.2 Unravelling the Networks

Once the super-network inter-vertex gaps have been filled in, the disparity map producer must descend a level into the vertices themselves. Since each vertex represents an entire network, this therefore involves examining the solution of the initial network reduction steps, and picking out the solutions.

This is achieved by placing a *pointer* at the source vertex of each network in turn, and following the sink links that were chosen to be of highest suitability from one vertex to the next until the sink vertex is reached (i.e. a vertex which has no further sinks). As each vertex is reached, its location within the match plane is read from its property list, and the disparity value calculated and placed into the disparity map. This proceeds until the sink vertex reached is the last in the chain.

5.2.7 And repeat...

It must be remembered that this process has now produced a one-dimensional array of values. Only one epipolar line from the initial image has been resolved. The whole process must be iterated over each and every line in the image.

Some performance can be gained by performing the process only on every *other* line, and interpolating between them if the matches are sufficiently different. This method works extremely well, and would have been left in the project, except it is considered a slight cheat. There is no reason this could not be used in a production system for optimisation purposes.

Since the production of a depth map is a simple matter of trigonometry, this step is left, since it is a well-defined one-to-one mapping, and unnecessary to demonstrate whether the system is working.

5.3 Disparity Error Analysis

There have been many sites with example stereo images. Most are artificially generated, since, as explained in the introductory chapters, the ray-tracing algorithm can be altered easily to also give the correct range information. If they have been artificially generated, or have already been processed by another stereo system, they may have been packaged along with their disparity maps.

If this is so, then a further stage can be added whereby these known disparity images are read in, and their values subtracted from the results in an absolute way, to give an absolute error. These errors can then be aggregated to give an overall error analysis, and assist in the fine-tuning steps.

5.4 Improving the Overall Matching

The match obtained is usually of sufficient quality to give a first approximation, and fair-quality dense range information. Successive steps could post-process the results, perhaps using feedback from other sensors, or human input. However, is it possible to improve the matching quality within the system, using the information available only to it? Would such improvements affect the overall performance?

The solution given in this thesis attempts to use every constraint available, and utilise every piece of data to the best possible result. There are still, however a few possible additions which present themselves, as well as some as yet unimagined.

5.4.1 Inter-Image comparison and smoothing

In the effort to increase the accuracy and operational envelope of the system, one continually runs into the problem of obtaining correlating data to substantiate match hypotheses. One possible source of correlating evidence can be obtained from temporally adjacent image frames. Say, for example, the system is running at approximately 4 frames per second, one can make the assumption that the system will seldom be presented with rapidly changing images. This means that, on the whole, the match planes will be similar in construction and appearance between frames. This is a slight extension of the smoothness constraint.

The smoothness constraint maintains that objects projected onto the image plane present a smooth intensity profile. Figure 5.11 shows an example one-dimensional slice across a match plane from an office test sequence. The slice is 20 pixels long, and shows the intensity across a coherent match, which happens to be across a vertical stanchion.

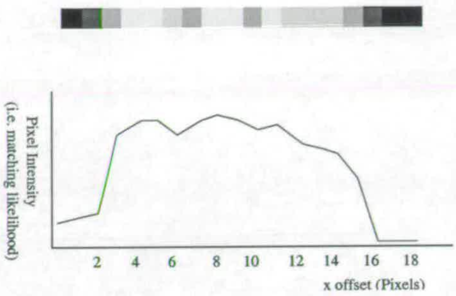


Figure 5.11: *A small section of a match plane, and a corresponding intensity graph.*

If one places the same slice from the 10 successive image frames, one obtains a plane, as shown in 5.12. One can see the profile given in Figure 5.11 in Image 1's slice. In successive image frames, which are in the receding direction (into the page), one can see that the matchable area diminishes towards the origin of the X axis. In the images, the stanchion is gradually moving to the left, as the camera pans past it. It can be seen that the move is gradual due to the high frame rate, and therefore, at any point on the surface, one can estimate its position with a certain accuracy given its spatial and temporal neighbours.

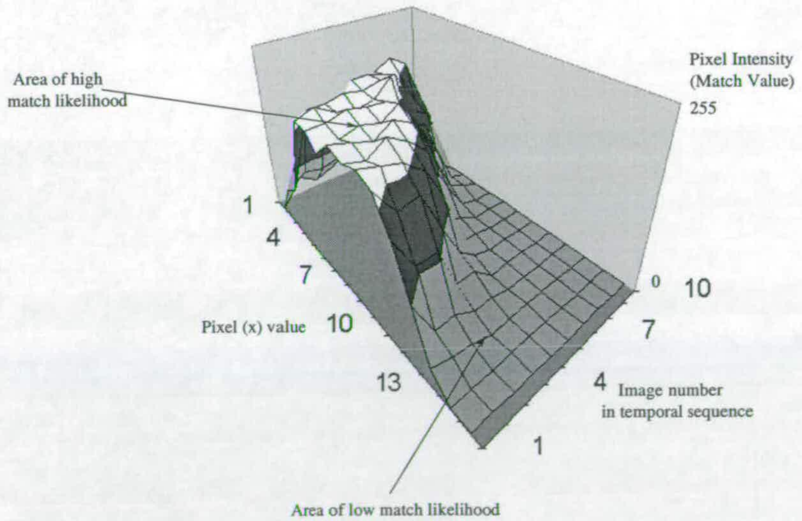


Figure 5.12: *A plane constructed from ten contiguous frames from the small section of a match plane, shown in Figure 5.11*

Solutions to the SCP, such as motion stereo depend upon this property of temporally adjacent image frames. If successive frames are too dissimilar, then all such monocular stereo algorithms cannot function.

It depends upon the environment in which the final product would be deployed to determine whether the usage of such information would help or hinder, on average. The main hindrance is the impact such a modification would have on runtime performance. The comparison of, say $knxn$ image planes would add a further kn^2 term to the runtime, so one must consider this carefully.

5.4.2 Adaptive Epipolar Line Finder

The main failing in such a system as this concerns the camera positioning. It is difficult to guarantee that the cameras will always be perfectly aligned, due to the sub-millimetre accuracies required. It is hoped that the data in the next section will convince the reader that exact Epipolar line matches are not essential, however camera decalibration is very easily done, especially in such a finely tuned system.

Horizontal alignment is often not important. If the cameras move laterally by some small amount, this has minimal effect on the results. Keeping the cameras vertically aligned, however, is a very difficult problem. They must be aligned exactly, since each raster scan line, which is typically less than a $\frac{1}{10}th$ of a millimetre, is to be compared with its exact counterpart in the other camera. Therefore, for close work, the cameras must be vertically aligned to within a $\frac{1}{10}th$ of a millimetre.

One possible solution is to detect the matching likelihoods emerging from the system. If the likelihoods drop below some threshold value, raster scan lines adjacent to the 'correct' scan line can be matched in parallel. Whichever scan line consistently gives superior matching results to the originally correct scan line becomes the new correct scan line, and the system continues.

This solution works only for translational errors. If the cameras are rotated in any way, with respect to each other, one must now consider another degree of error. Since the raster nature of the image surfaces allows much optimisation, incorporating the necessity to calculate a set of angled epipolar lines often proves too computationally expensive, and many systems simply assume that rotational errors will not occur. Another way to tackle this problem is discussed in Chapter 7, in the section on further work.

Chapter 6
Results

6.1 Introduction

The system was tested on a number of diverse datasets. Each has been chosen to specifically test each of the weaknesses of the pixel-based stereo solution, i.e. featureless regions, parallax, occlusion, misalignment and difficult lighting conditions.

The first is artificial in nature, and acts as a reference set to work from. The others are natural. The artificial dataset is a standard image, of a corridor scene. The natural scenes are taken in an office environment, and the outside world, to test the system under artificial and natural lighting conditions.

Each section will describe a scene, and the particular difficulties arising from it. Each of the major problems posed by the scene will be dealt with independently, and the system's performance, expected result, error and cause will be discussed. The sets are referred to as the *Corridor, Boxes*, and *Cement Truck* datasets.

6.2 The Corridor dataset

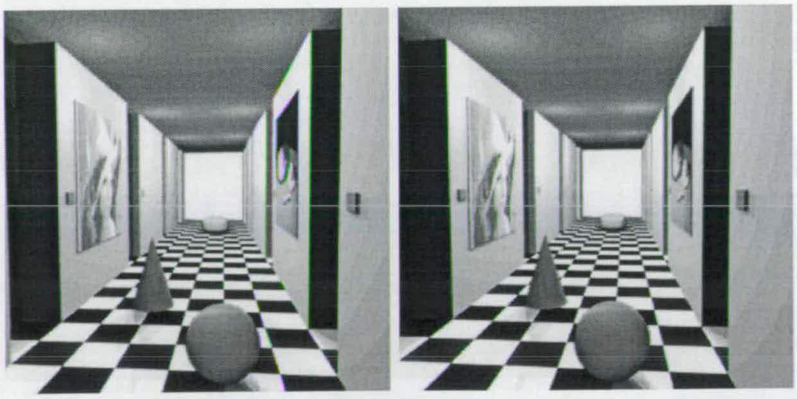


Figure 6.1: *Left and right images from the Corridor dataset.*

6.2.1 A Discussion of the Scene

The Corridor data set, is a standard scene used much in the field of Stereo Vision. The images are available in many formats, however the images displayed here are TIFF format 24-bit 256x256 grey images. They have been converted to PGM format, which is an 8-bit grey-level representation. The images are a good test because they have a good clean aspect of the scene under inspection. The scene contains a view down a corridor. On the walls of the corridor are protrusions, and cut-outs where light switches and side-corridors add a little complexity to the enclosing textures. The corridor recedes into the image, with a picture on the left wall to add texture to an otherwise plainly rendered scene. Also to add complexity, there is a sphere, cone and torus placed at varying distances onto the chequer-pattern floor.

This dataset, therefore, contains no lighting anomalies, due to the lighting model, nor does it contain features of reflectance or transparency. It also contains very little occlusion. It does, however contain many areas of featurelessness. These areas, however, are well bounded, and therefore do not suffer the parallax, alignment or occlusion problems of other images.

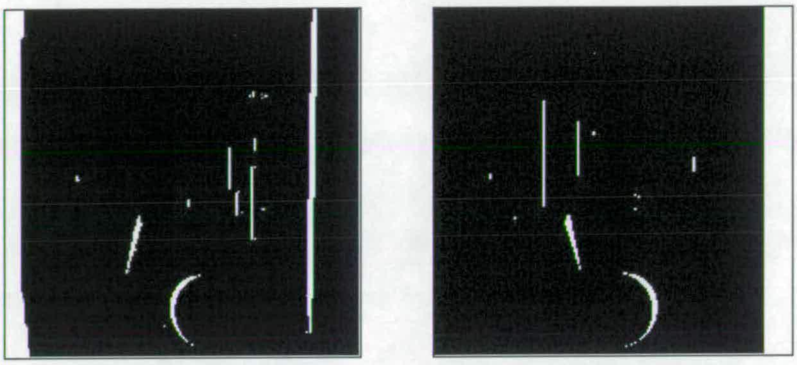
The stereo pair come with the associated range information pre-computed. It is this data which allows the researcher to compare the system results with the accurate values to determine net error. This pre-computed range information will be used to test performance and will be referred to as the *ground truth* or *correct* result.

The stereo pair also come with a number of other pairs, which are exact copies of the originals, with varying amounts of noise added. Again, these other images are an excellent addition, and allow the system to be assessed qualitatively by automatic, and quantifiable means.

6.2.2 Processing Analysis

The corridor dataset is typical of computer generated stereo images. It contains large planar surfaces with little texture. The quality of its lighting is remarkably good, possibly using sub-pixel radiosity. This reduces the number of artifacts caused by the raster nature of the image, and reduces the number of false positives.

Upon the 'floor' of the image, there are three small objects. There is a sphere, torus (far distance) and a cone. Their presence is to introduce a small amount of occlusion into the centre of the image. Figure 6.2 show the amount of occlusion present. White areas show those parts of each image which do not appear in the other. They cause the flow network solution a particularly difficult problem to solve, since it cannot match that which has *no* match, yet it cannot tell in advance which areas are unmatchable, and bypass them.



a) b)

Figure 6.2: *Occlusion maps for the Corridor scene. a) Left-hand occlusion map b) Right-hand occlusion map*

Figure 6.3 shows the effects of occlusion at the edge of the spherical object. As can be seen, slightly more floor is visible around the curve of the sphere in the right hand image. Normally, this may not be a problem, however the floor itself is textureless. Taking a horizontal section from the image, across the centre of the area of interest (line 223), gives the resulting section of match-plane as shown in Figure 6.4

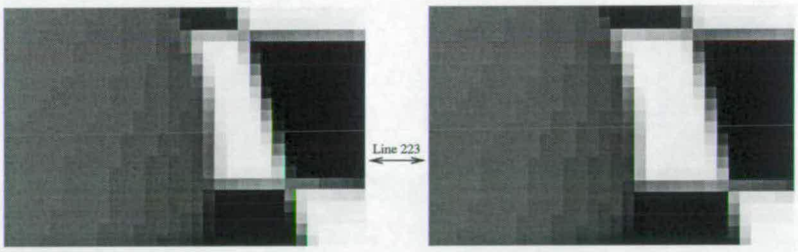


Figure 6.3: *Close-up of the disparity close to the edge of the sphere*

After processing, this part of the image is converted into maximum likelihood network nodes, and solved. Figure 6.5 shows the area during network path processing. The darker grey pixels show possible candidates. The light grey pixels show those candidates which have been chosen after the final super-network reduction. The small conglomeration of pixels in the centre of the image show the maximum likelihood pixels from the match plane which are chosen. The white line through this area shows the where the *correct* selections of pixels should lie. As can be

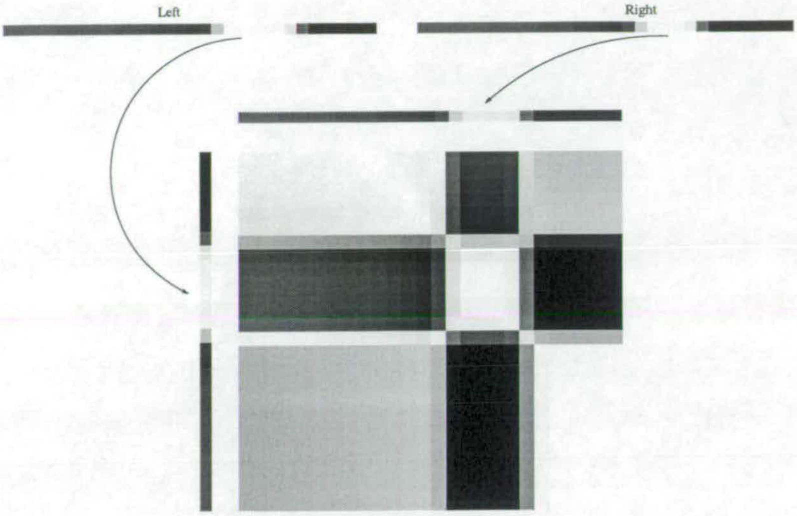


Figure 6.4: *One section of epipolar line 223 which passes through the occluded region, and its corresponding section of match plane.*

seen, the lighter grey pixels in the central group have been chosen incorrectly!

The question must be "why did it choose the wrong path?". The answer to this is simple. The network reduction step chooses the *best* path, based upon pixel intensities, and as can be seen from Figure 6.4, it has done so. The problem lies in the fact that due to occlusion, and the overall featurelessness of the area, the reduction step is unable to tell the difference between a featureless object viewed under parallax, or a featureless object occluded. Figure 6.6 demonstrates how the profile of the epipolar lines are identical for the two cases.

As can be seen, the solver decides that the area is simply a parallax issue, and solves it accordingly, which is, of course, incorrect. It should have refused to match the central portion and marked it as an occlusion. How, however, was it to know that it was an occlusion?

Additional logic, whereby the solution is checked with previous and succeeding epipolar lines, could be used to perform a rough guess as to whether the matcher should solve as an occlusion or parallax. The computation necessary, however, is complex, and not always intuitive. There was insufficient time to fully work out all of the logic and implement it for this step, and was omitted. It is discussed in the further work section later in this Thesis.

Other than at areas of occlusion, the matcher performs well, generating a result which agrees

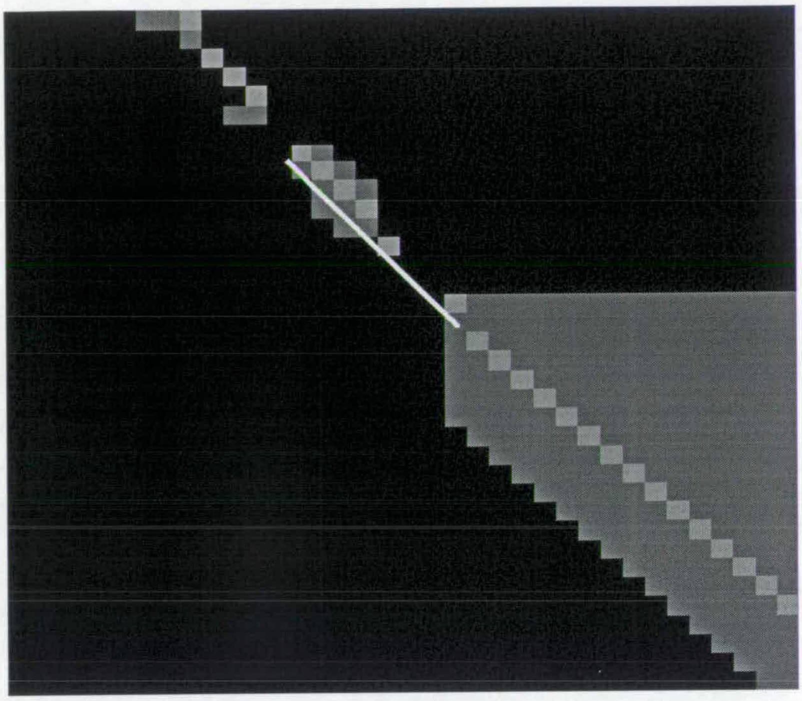


Figure 6.5: *The correct and chosen paths through the occluded region of epipolar line 223 (The ball) in the Corridor data set.*

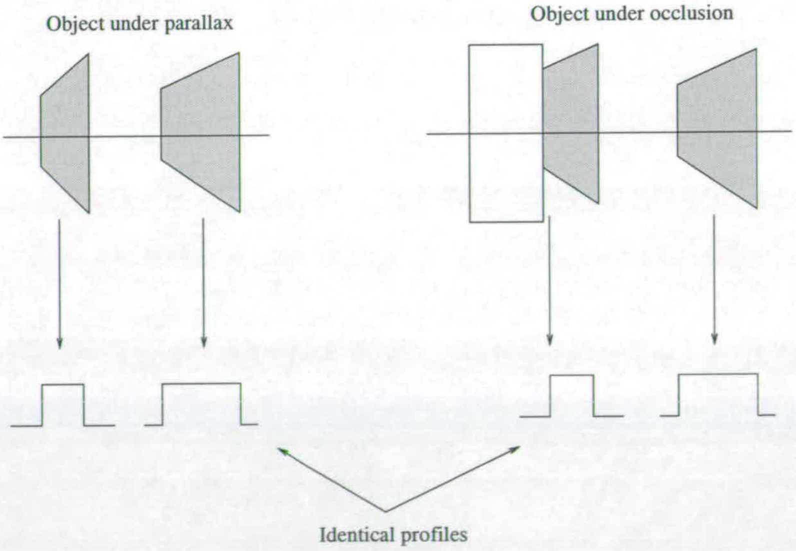


Figure 6.6: *The epipolar profiles of an object under parallax distortion and occlusion*

with the ground truth disparity provided with the dataset in over than 70% of the image area (within an acceptable 1 disparity level of error). The result is shown in Figure 6.7 below.

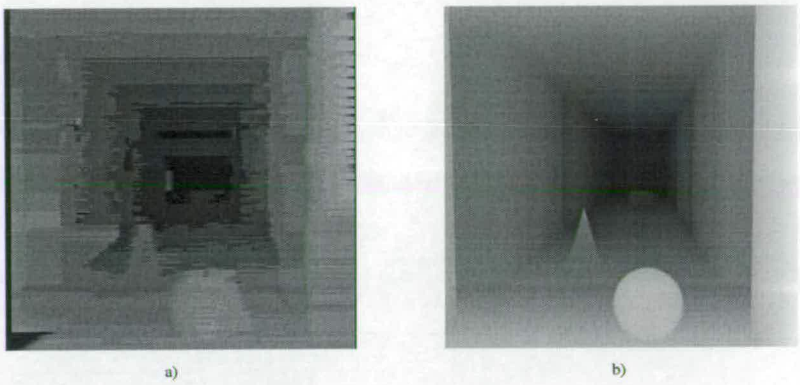


Figure 6.7: *Final solution to the Corridor data set (a), and the 'correct' solution (b).*

As was mentioned earlier, the image has been rendered to sub-pixel accuracy, however in areas of adjoining colour, say at wall/ceiling boundaries, some anti-aliasing has been performed. This anti-aliasing, occasionally, can cause the *best* path to be offset by a pixel. Anti-aliasing has the effect of blurring the boundaries of objects, and hence blurring the location of the source and sink vertices in the networks. Because of the inter-pixel peer-to-peer nature of Dykstra's algorithmic network reducer, whereby a pixel may be chosen based upon its intensity and/or whether a previous pixel path was chosen, the moving of the source pixel of a network can cause a totally different path through the network to be chosen. This is especially true in areas of featurelessness, such as the walls and ceilings in these images, since all paths through have exactly the same 'goodness'.

When all of these are factored in, an image difference between the correct solution and that provided by the stereo solver can be performed. Figure 6.8(a) shows the output, as given by the system. Figure 6.8(b) shows the correct solution, as provided with the dataset. Figure 6.8(c) gives a difference between the two solutions. Darker areas show where difference is lower, and hence the solver's solution was closer to the intended result. The graph in Figure 6.8(d) gives the histogram of pixel errors.

The depth map resulting from this dataset contains 12 disparities ranging from 1 to 12, hence the blocky appearance. The correct solution provided with the dataset has been rendered to such an accuracy that it contains 203 discrete disparity levels. This is because the ray tracer used to create the solution has not been configured to report only integer value disparities, and

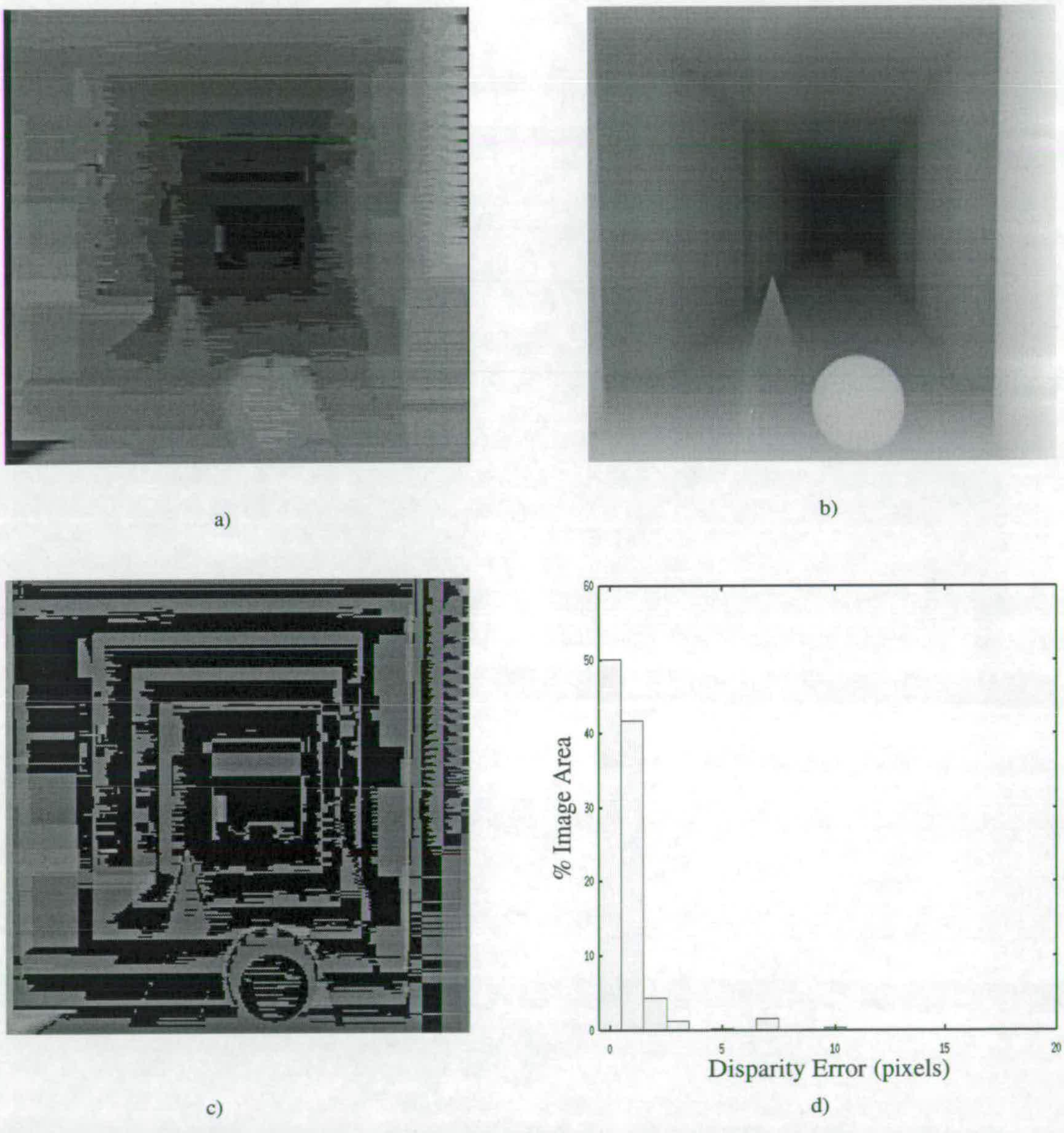


Figure 6.8: *A Corridor Solution compared with the correct solution. a) Stereo Solver Output b) Correct Output c) differences between a and b, amplified to improve visibility. Mid grey is error level of one. d) Histogram of errors*

has scaled the results by 21.25 to make use of the full 255 grey level range available in the TIFF image format. Our stereo matcher, however, is presented only with discrete pixels, and can therefore only report integer disparities.

Figure 6.9(a), shows two objects, when projected onto the two recording surfaces having a disparity of 8 and 9 pixels. They are quite close to the cameras, and may be a few centimetres apart in depth. However, Figure 6.9(b) shows two objects which are far from the recording planes, and therefore have only small disparities, say 1 and 2 pixels. As can be seen, these objects may be separated by more than a meter, dependent upon the baseline.

Hence, it is not always possible to represent every depth. For example the midpoint depth between Pa and Pb in Figure 6.9(b), (which should have a disparity of 1.5 pixels), would be rounded to 1 or 2, due to the discrete nature of the images.

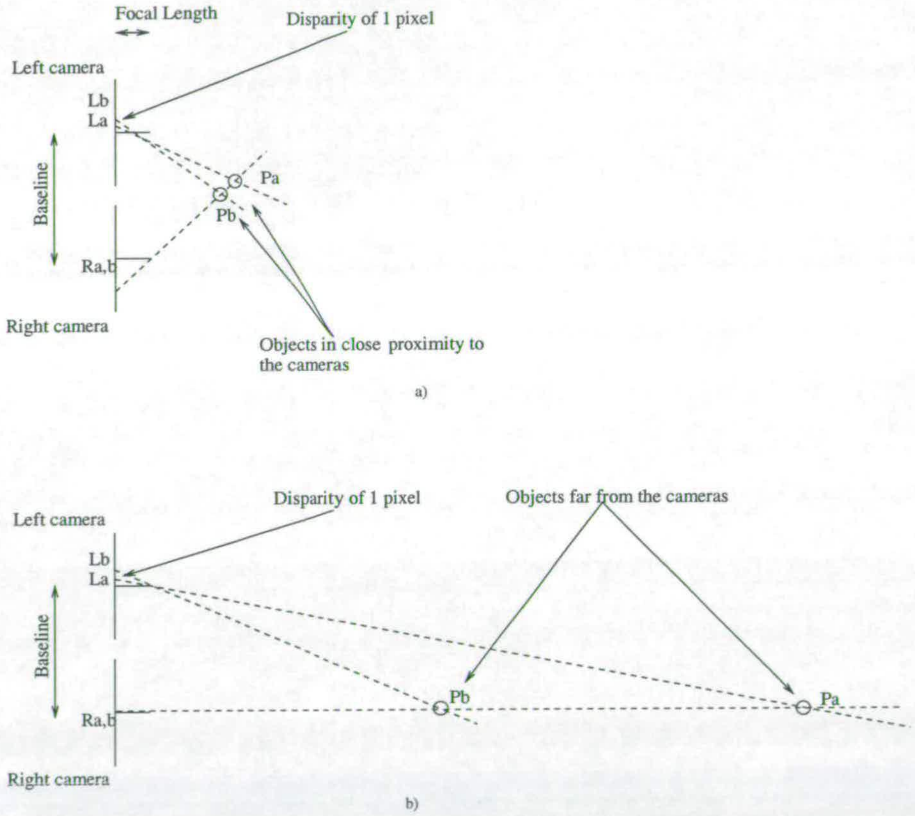


Figure 6.9: *How distance from the camera affects the recordable depth.*

Figure 6.10 shows the distances of equivalent disparity. Any object that is not located exactly at these distances, has its disparity rounded to be on one of them. Ray tracers do not have this problem. The ray tracer will report the non-integer depth correctly, recording this non-integer depth at that point in the result.

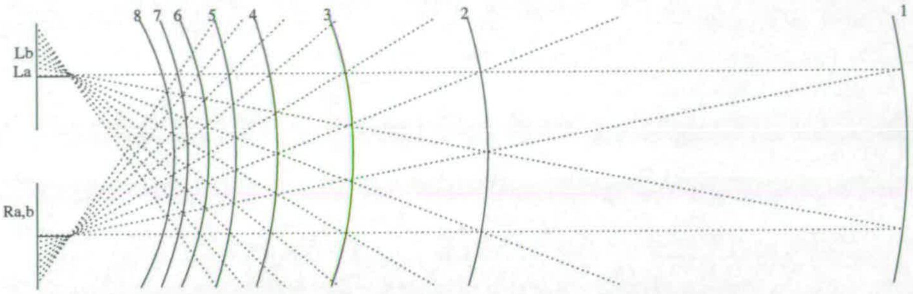


Figure 6.10: *Distance 'shells' of equivalent disparity.*

Therefore the last stage, before production of the 'correct' result image, is to perform the conversion, as given in Equation 6.1 upon each pixel $P(x, y)$ in the image.

$$GreyLevel_{P(x,y)} = (Disparity_{P(x,y)} * 21.25) \tag{6.1}$$

To recover a correct solution, with which to compare the results in a manner closer to that which the stereo solver should output, the correct solution was re-mapped or *quantised* into a smaller number of depth bins (12). This was done by performing the inverse of the equation given in Equation 6.1. The operation for each pixel, $P(x, y)$ in the disparity map is given in equation 6.2

$$Result_{P(x,y)} = abs(SolverOutput_{P(x,y)} - (\frac{CorrectOutputP(x, y)}{21.25})) \tag{6.2}$$

Figure 6.11 shows the effect of quantising the correct ground truth to integer disparity levels, and then performing the equation 6.1 to stretch the grey levels back to allow them to be easily viewed in a document such as this. As can be seen, much depth data is lost, however this represents the optimum depth map, one would expect a stereo solver such as the one given in this thesis would output.

This would suggest that the matcher is performing fairly well, using a pixel-based approach. It

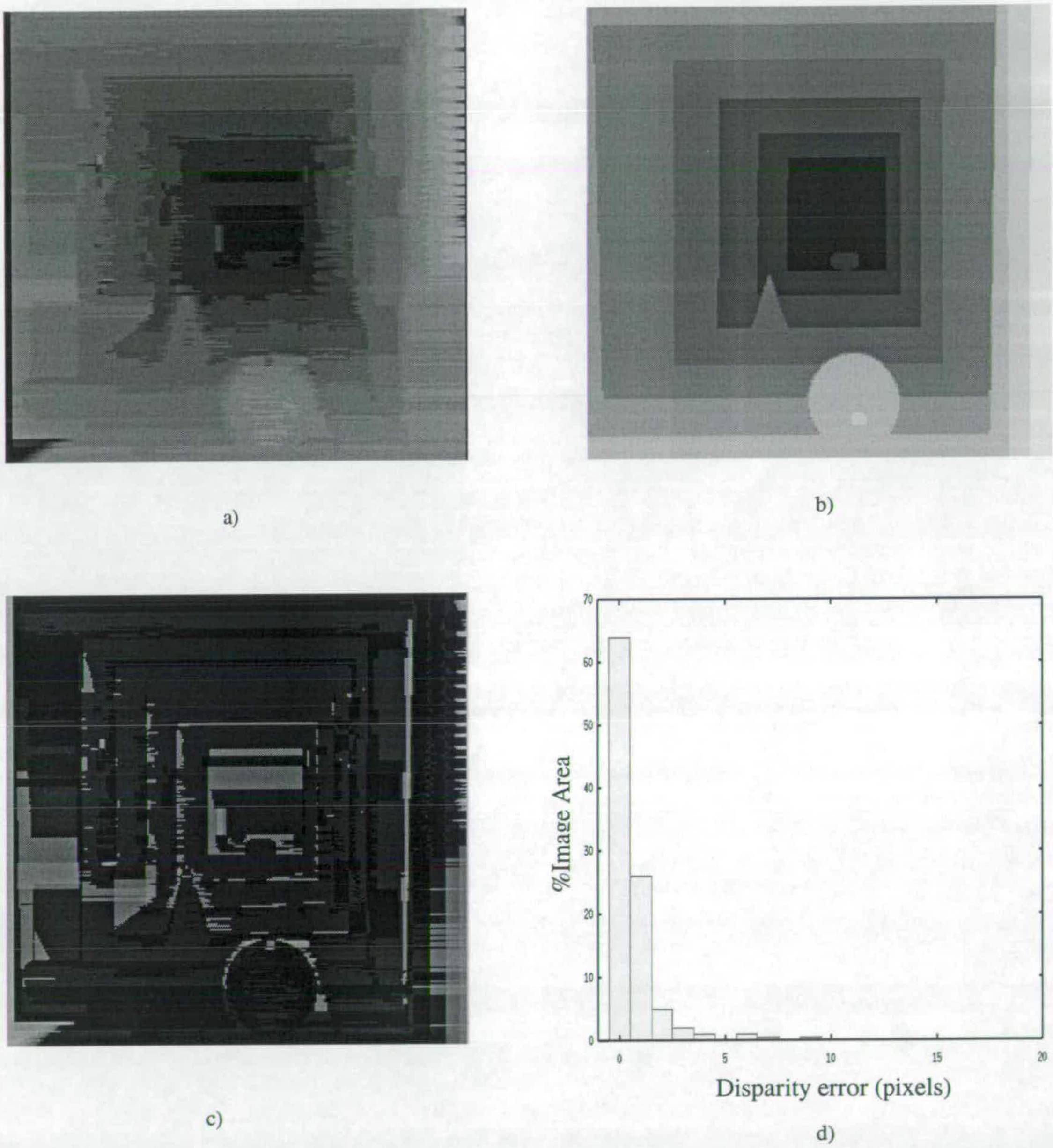


Figure 6.11: *A corridor solution, compared with a quantised version of the correct solution. a) stereo solver output b) quantised correct output. c) pixel delta between a and b d) histogram of errors*

would furthermore suggest that a better solution rests only with further extrapolation, or vertical interpolation. When compared against the quantised result, the matcher is able to match 92% correctly to within an error of 1 disparity level, which is permissible.

The runtime of the solution is fair. At just under 26 seconds to complete the image on a Sun Sparc 10 workstation, with little optimisation, and disk I/O writing the solution, this shows that the method is usable, and, if some of the optimisations mentioned above included, practical on images of this simplicity. Several techniques were suggested and implemented to reduce the runtime of this dataset. The most dramatic was the block-removal step, described earlier. This reduced the runtime down to under 14 seconds; however it introduced some anomalies, especially around the fine detail points on the cone and sphere.

6.2.3 Adding some Error

The Corridor Dataset comes with three additional image pairs. These pairs are identical to the original scene, except that they have added noise. The noise is gaussian-distributed, and modifies the intensity values of the pixels by up to a certain amount. The three pairs have noise variances of 1, 10 and 100.

As is expected, as the added noise increases, the accuracy decreases. Since this solution is solely based on pixel intensities, and their positions relative to each other, adding noise throws off the matcher significantly.

To counter this, the matcher is allowed to "skip over" up to two pixels which do not match. This value is configurable, but affects runtime due to the increase in search space. Therefore, the drop-off in accuracy, as noise is added is not as sharp as could be expected.

Figure 6.13 shows the relative errors resulting from adding noise to the datasets. As can be seen, the drop-off in accuracy resulting from the addition of noise is very marked. The final dataset achieves an accuracy of 23% only because the results are almost completely random, and some correct hits are bound to occur!

This demonstrates the dependency that a pixel-based solution has upon clean, calibrated data. Thanks to the findings of these tests, the images in the succeeding datasets were colour-reduced to 64 grey-levels (6-bit) from their original 256 (8-bit values). This is because the random noise added to the images, is so low that it affects only the bottom two bits of data, thereby reducing

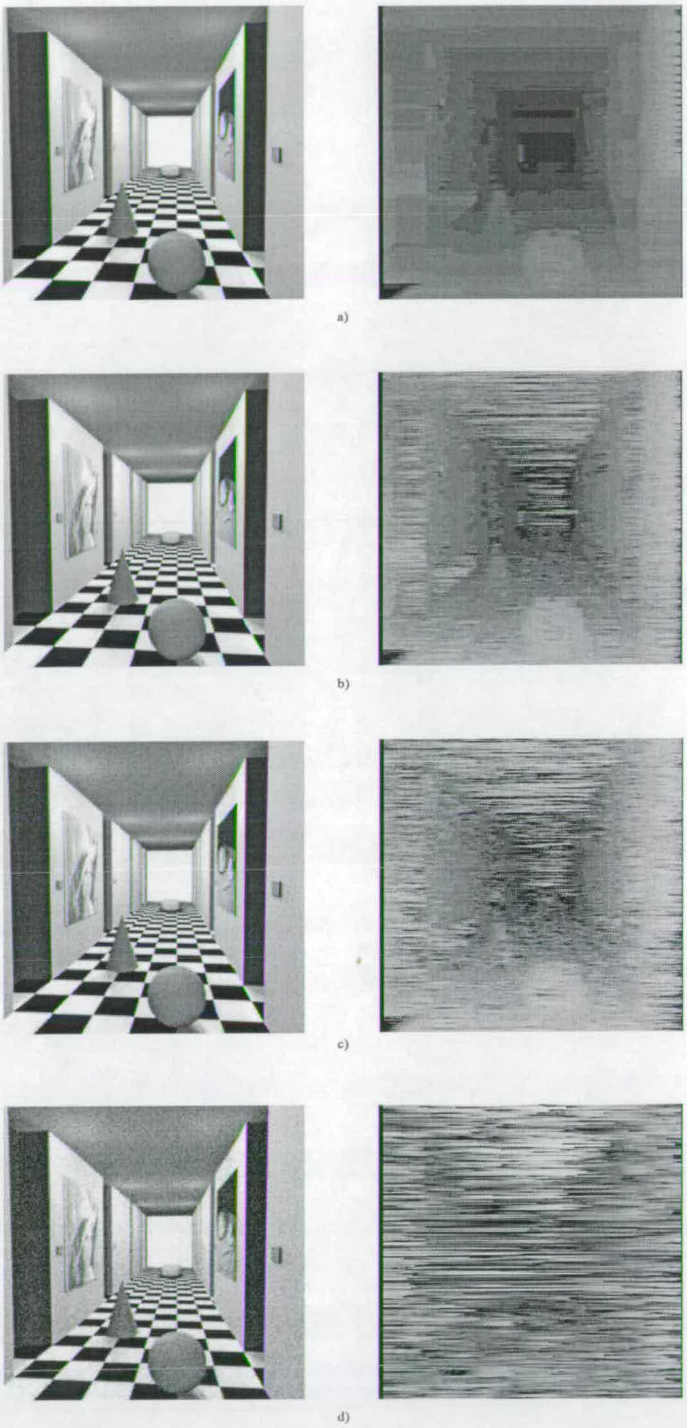


Figure 6.12: *The Corridor dataset with added noise. a) Original Scene with resulting disparity map. b) Noise of variance 1 added, and resulting disparity map. c) Noise of variance 10 added, and resulting disparity map. d) Noise of variance 100 added, and resulting disparity map.*

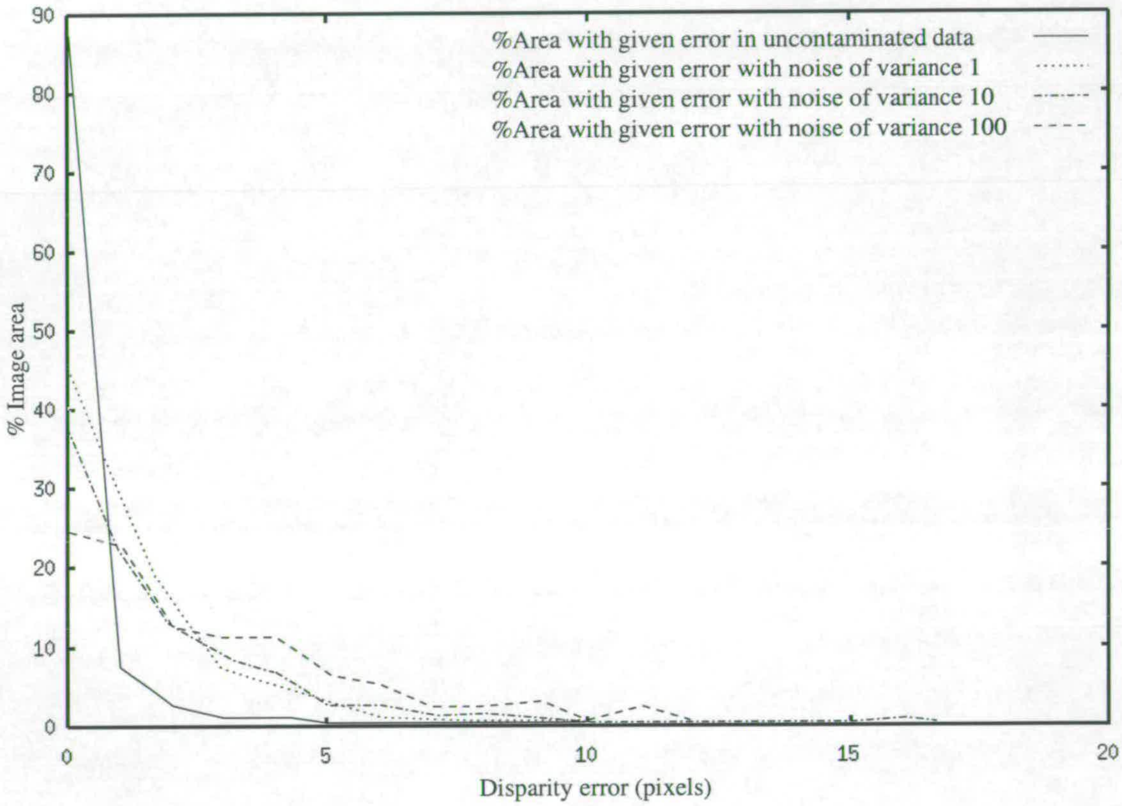


Figure 6.13: *Distribution of errors across the Corridor data set with added noise, compared against the quantised result.*

the effective range to 64 discrete levels. This error was measured by taking 4 natural-light images in an external environment in quick succession, with the camera clamped to a baseboard. The differences between each image were calculated using image differencing (delta operations), and found to be +/- 4 grey levels on average. This error may be introduced by the camera, lens, or slight lighting differences.

6.3 Boxes Dataset

6.3.1 A Discussion of the Scene

This first, *natural* dataset is cluttered compared to the corridor scene, and contains many *annoyances* for a stereo depth solver. The scene is 'noisy', since real cameras are used. The data is far from perfect, and the lighting in the office has a very vertical spot component, when compared to its ambient component. To complicate matters, there are many areas of occlusion and some featurelessness.

The images were taken with a Sony DSC-S50 Digital Still Camera at a resolution of 640x480, clamped to a mounting which could slide up to a metre horizontally. The camera was configured to take an exposure of greater than 1/25th of a second to overcome fluorescent light flicker.

The images were then reduced to grey-scale, and their centre area merely cropped to 256x256. No resizing was performed, since this would introduce aliasing errors. The camera baseline is one metre. The boxes are approximately 3.5 meters from the camera, and the wall beyond, 2.0 meters further still. The lights are of a ceiling-mounted strip-bar type, mounted in groups of four within chrome diffusers. Approximately 16 bulbs are lighting the scene directly from above.

Figure 6.14 shows the stereo pair. As can be seen, it is an extremely difficult scene. With such little colour variance, and so much clutter, the system, understandably, has problems with many areas of the image. Of the three scenes, this is the most difficult, chosen specifically to counterpoint the simplicity of the corridor dataset.

It was expected that the system would not be able to resolve much from this scene. Large areas of similar colour tend to generate large networks, and would therefore have an extremely long run time, while the correct network routes were reduced. As shown in the Results section, both the runtime and the results proved to be unacceptable for a real-time stereo system. They were,

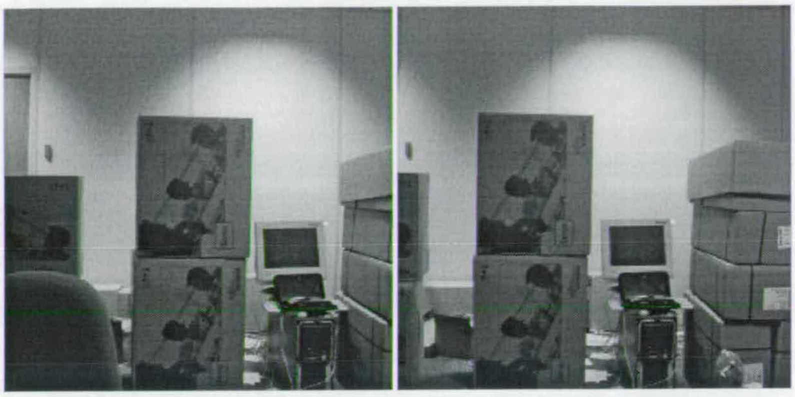


Figure 6.14: *The left and right images of the 'Boxes' dataset.*

however, better than expected.

The boxes are traditional cardboard brown, and the wall behind, a slightly lighter brown. There is very little colour variance across the image in fact, which is guaranteed to give a pixel- based solution problems. All of these were outlined earlier, as most likely to cause the algorithm problems. This scene, however, does contain a moderate range of disparities, varying from the chair in the immediate foreground, to the wall further away.

While a variety of disparities is good for object distinction, it does introduce a large amount of occlusion, especially over featureless areas, which, as stated earlier is an exceptionally hard problem for spatial-based stereo systems. The floor has some papers littered over it, in some areas to introduce some texture, however, as can be seen, in most cases the occlusion completely obliterates that data in one or other image.

So one can conclude that the scene, setting, and lighting all conspire to make tricky scene for a pixel-based stereo system.

Unfortunately, natural images, unless processed by some other system, do not come with range information, therefore *correct* depth maps must be created by hand, by sampling the scene using a tape measure, or by comparing the images by eye to obtain pixel offsets. To create the correct maps in the following two scenes, the depths were measured with tape measures, and a CAD package used to map the depths onto the objects within the images. In order to test the inverse trigonometry, the stereo pairs were inspected individually, and their offsets recorded manually.

6.3.2 The Results

The results for this dataset were very encouraging. There are large recognisable areas where the disparity has been recovered, as can be seen from Figure 6.15

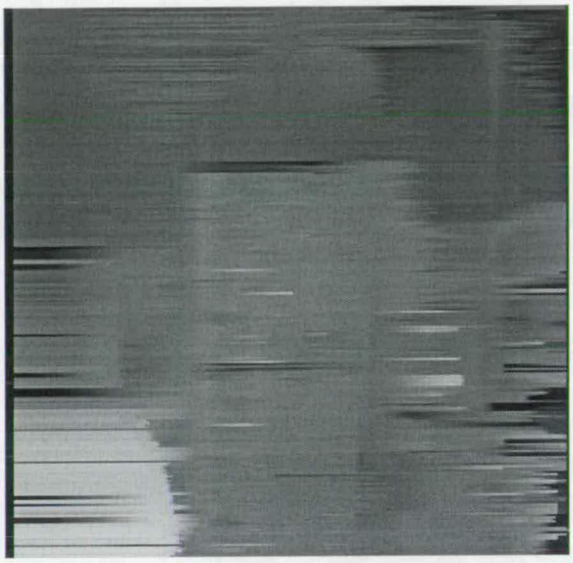
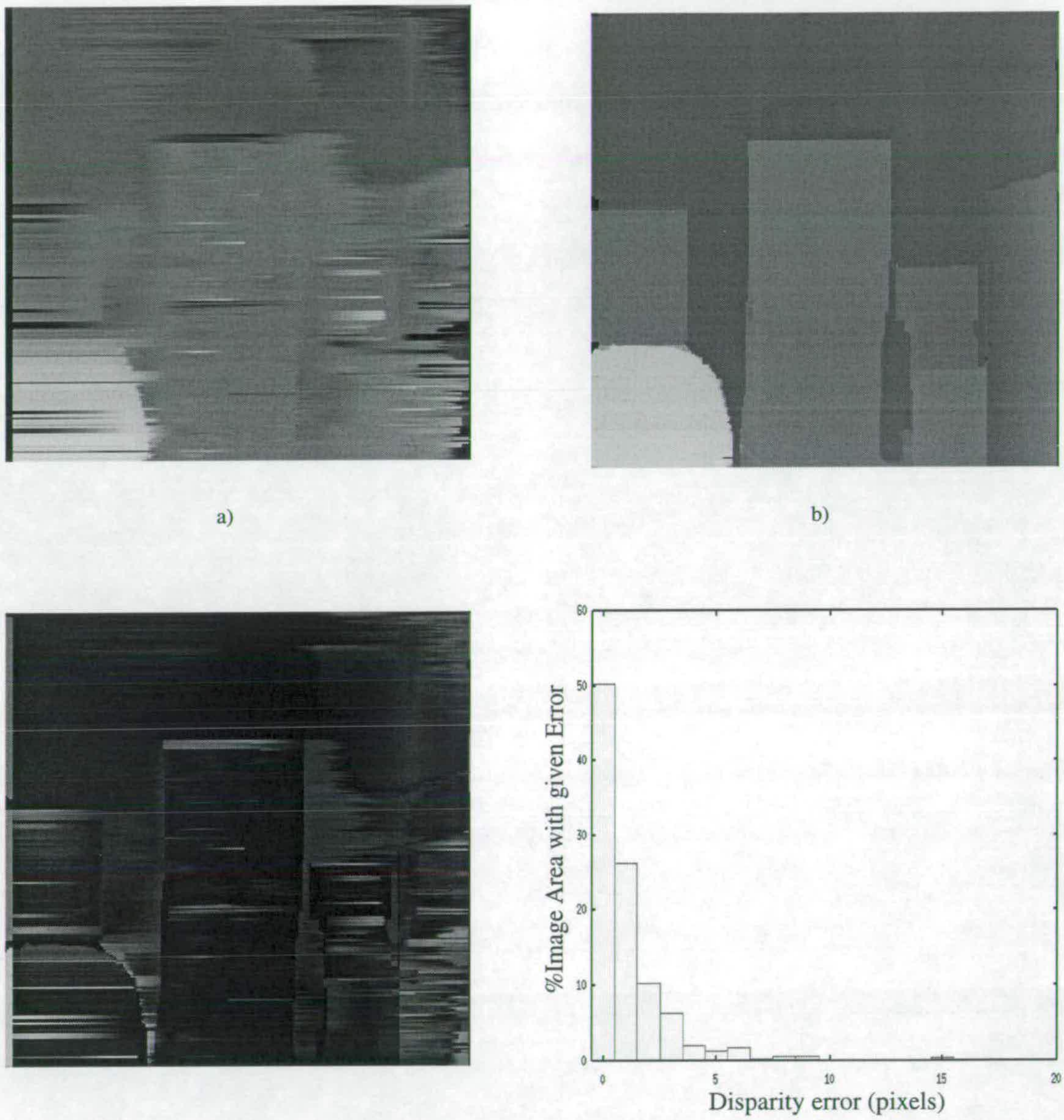


Figure 6.15: *Disparity map for the 'Boxes' dataset.*

An example of an area well recovered, is that of the chair in the foreground. Only a slight curve of the chair's back is visible in the right-hand image, however this is enough for the system to estimate the disparity of the whole back, and (where possible) mark it correctly. The boxes, can similarly be seen to be at the same rough disparity, indicating that the system, once again, has managed to recover the depth, moderately consistently.

Figure 6.16(a) shows the recovered depth, as output by the solver. Figure 6.16(b) contains a depth map generated using a tape measure and a CAD package. The distance from the baseline to 10 points within the scene was measured, and the expected pixel offset due to that offset was calculated. These values were then imported into a CAD package. Figure 6.16(c) shows the result of performing a difference operation between parts a and b. As can be seen, the boxes, and wall are fairly well recovered. Figure 6.16(d) shows the errors in terms of image area. As can be seen, the accuracy of this solution is far lower than the corridor set. While the corridor set was at least 75% accurate, this set can be seen to have only 64% of the image area correctly matched, with a further 15% with an error of just one pixel, giving an acceptable area of 75%. Considering the complexity of the scene, this is, quantitatively at least, a fair solution.



a)

b)

c)

d)

Figure 6.16: *A Boxes solution, compared with a manually generated depthmap. a) Stereo Solver Output b) Correct Output c) Pixel delta between parts a and b, d) Histogram of disparity errors*

The errors within the images come from one of two main stereo problems, occlusion or mismatching. Occlusion within these images is a big problem for the wall behind the boxes. There were few examples of mismatching that were not induced by occlusion error, as the camera was of a sufficient quality to ensure moderately faithful images, despite the lighting differences, and there were no non-lambertian surfaces. The following two examples are places where the matcher was unable to cope with the lack of information due to occlusion.

6.3.2.1 Error due to Occlusion

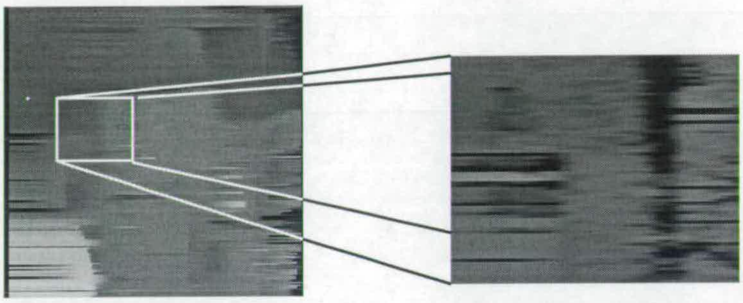


Figure 6.17: *One section from the 'boxes' dataset which displays an area of error due to occlusion.*

Figure 6.17 depicts a segment of the disparity solution to the left hand side, between two boxes. This area has been used because of the large error in the inter-box space due to occlusion.

Figure 6.18(b) shows the same segment, however using a measuring tape and inverse trigonometry the approximate disparity has been drawn over the boxes. Figure 6.18(c) shows the result of doing a pixel-by-pixel difference between the two images. As can be seen, at the top left of the image, the background wall is measured accurately in the most part. The boxes, however, have some significant areas of error. Figure 6.18(d) shows the histogram breakdown of the errors.

Assuming that an error of +/- 1 pixel is acceptable, 40% of the area is correct, with roughly 60% within tolerance. If this system is to be used commercially, this is a poor result. Why is this?

Figure 6.19 shows the intensity graph across this occlusion area, while Figure 6.20 shows these two graphs when superimposed. As can be seen, these two graphs are very similar, and it is clear that a solely value-based matcher would group together the similarly valued zones.

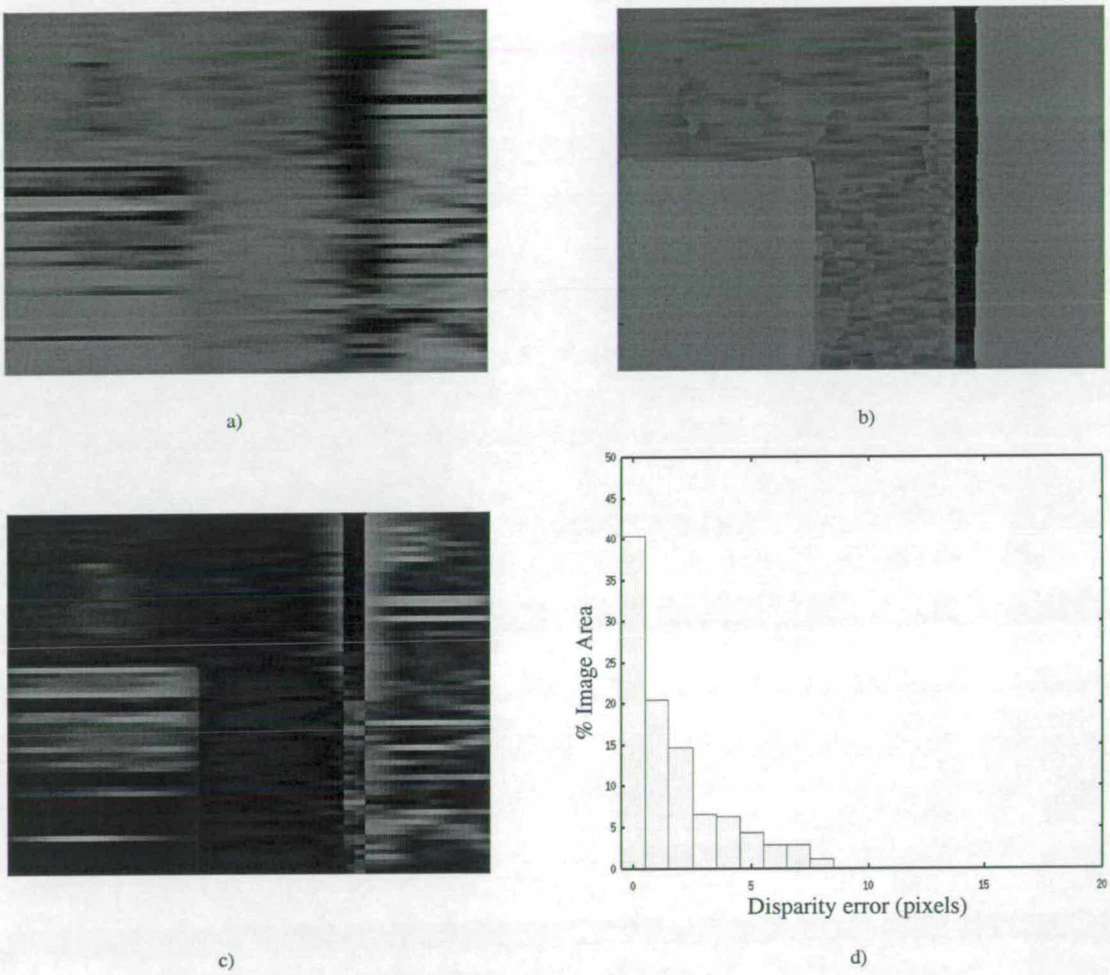


Figure 6.18: *An analysis of the error within this section 'boxes' dataset. a) The disparity values. b) The estimated correct disparity values. c) The pixel difference between a. and b. d) The distribution of error within this area.*

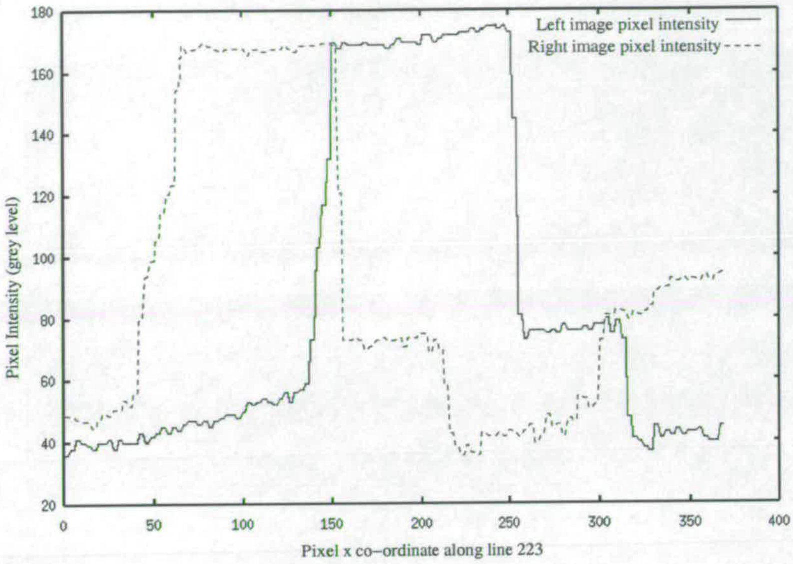


Figure 6.19: *Intensity graphs across the occlusion area.*

However, the central spikes represent *two separate* parts of the back wall, due to the effects of occlusion. When the matcher incorrectly matches them, therefore, they are assigned erroneous depth values.

6.3.2.2 A gross mismatch error due to Occlusion

Figure 6.21 shows another section of the image which suffers greatly from the error caused by occlusion. It is clear that the depth of the section of back wall, visible to the right of the box is incorrectly recovered.

Figure 6.22(a) shows the area under scrutiny in this section. Figure 6.22(b) shows what the area should be, if the matcher were to recover the depth accurately, and correctly. This ground truth section was created by hand, comparing the two views, and using a CAD package to recreate the ground truth map. When the recovered and correct sections are compared, they result in the difference map shown in Figure 6.22. The error shown here has been multiplied severalfold to render it visible. The graph in Figure 6.22 shows the area of section covered by errors. As can be seen, this area is very badly recovered. Less than 20% of the area is identified correctly. Approximately 30% is recovered with only a single depth-unit of error. the upper right corner of the image (i.e. the back wall) accounts for almost all of this. More than 50% of the area, however, is recovered with errors well outside tolerance for a scene such as this.

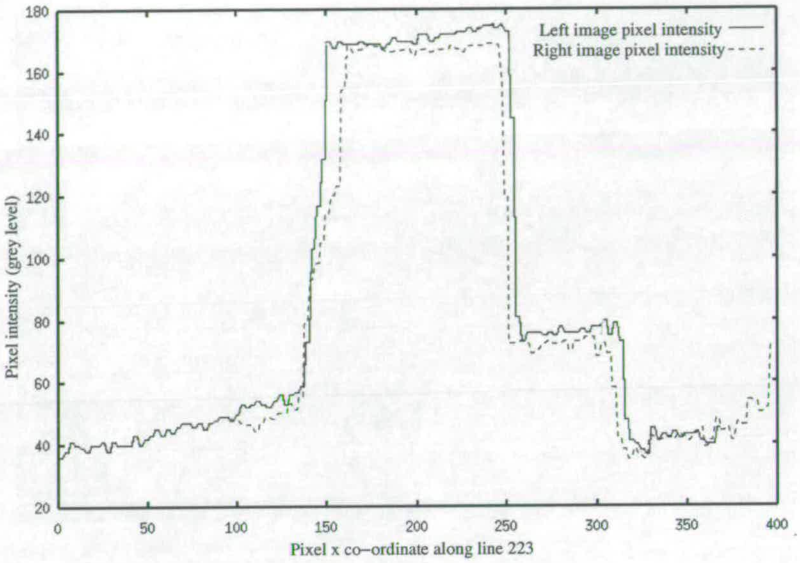


Figure 6.20: *Intensity graphs across occlusion area, superimposed.*

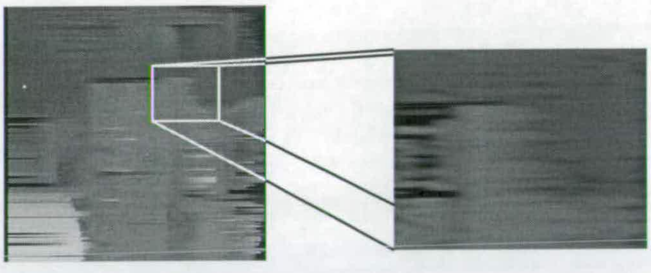


Figure 6.21: *Another section from the 'boxes' dataset which displays an area of error due to occlusion.*

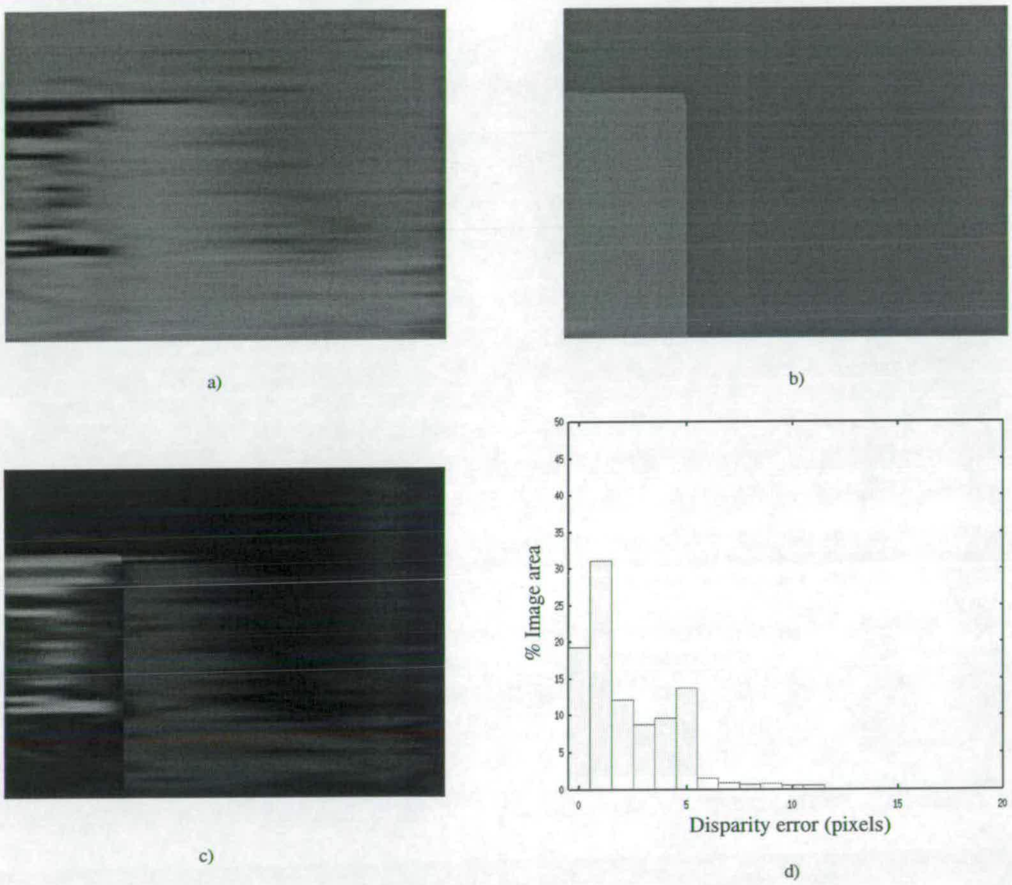


Figure 6.22: *Analysis of another area of error within the boxes dataset. a) Recovered area. b) correct result. c) Difference between result and correct result. d) Graph of error against percent of image area.*

The reason for this amount of error is shown in Figure 6.23. A top aspect of the scene is shown in part 6.23(a). The correct depth graph for an epipolar line through the area is shown in 6.23(b). Figure 6.23(c) shows the actual depth, as recovered by the system.

Why is this? There are two arcs of dashed lines in Figure 6.23(a). These lines show which sections of the back wall are visible within each image. As can be seen, it is a different section for each. However, because the back wall is featureless, both sections (although different sizes) are identical, and therefore matched, exactly as in the first featured error area.

What makes this interesting, however, is that when the depths are recovered, the assumed parallax distortion changes the depth of the wall such that it seems to gradually project 'forward' in the image, and adjoin the front face of the box. To a postprocessor, this would make the box seem more of a trapezoid in cross-section, with one side coloured as the back wall.

It is difficult, once again, to see how to deal with such situations. The only solution seems to be to refer to another piece of evidence, such as a temporally adjacent frame, or a spatially adjacent epipolar line. Unfortunately, in this case, the occlusion has caused a serious mis-matching to occur, and hence the recovered depth is grossly wrong. In a scene with more textual data, this may not be quite so severe. However, in the study of many natural datasets, featureless, or fine- detail repetitive patterns are very common, and until this particular problem is solved, a matcher such as this, will always fall into these traps.

6.4 Cement Truck dataset

6.4.1 A Discussion of the Scene

This final dataset tested is a “middle-ground” in difficulty. It is a pair of images of a cement truck, sitting outside a warehouse, with a stone building in the background on the left. In the immediate foreground, to provide some clear relief, and, of course, some difficulty, there are a number of blue chemical barrels. The images themselves were taken on a partially overcast day, such that the light striking the scene would be non-directional and uniform.

The dataset contains a number of occlusions, similar to the boxes, however, like the corridor, it contains large areas of clear, uniform colour. All surfaces within the image are Lambertian in reflectance, and there are few areas where adjoining objects at different disparities have similar colours.

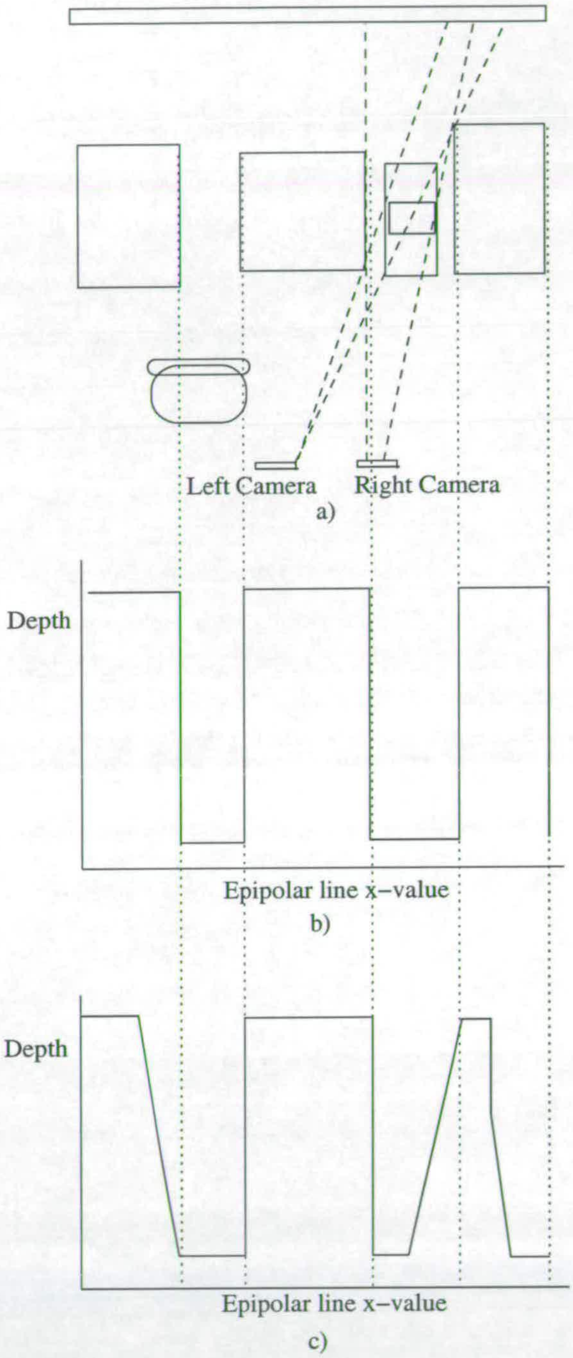


Figure 6.23: *The Boxes Dataset. a) Top-down schematic. b) Correct Depth. c)recovered Depth*

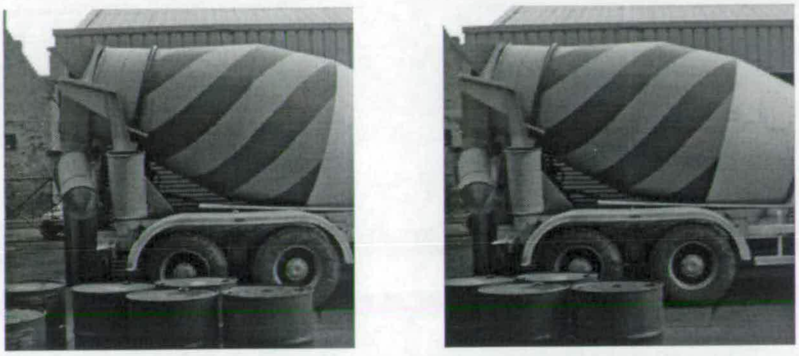


Figure 6.24: *The 'Cement Truck' dataset.*

These images, like the boxes dataset, are taken in the real-world. They were taken with a Sony Cybershot DSC-S50 Digital Camera, at 640x480 resolution. The top left-hand 256x256 corner of this image was taken from the left and right views to form the stereo pair.

This dataset highlights the performance of the system in another of its possible deployment arenas, i.e. the outside world. While the boxes dataset is quite cluttered, this dataset is relatively sparse.

6.4.2 The Results

Considering that this dataset is a middle ground in complexity, the results for it are quite disappointing.

Figure 6.25(c) shows the difference between the correct disparity map, as shown in part 6.25(b) and the result from the system in 6.25(a). The truck itself, the barrels and the shed behind are all rendered satisfactorily, however the tarmac in the lower-right, the entry-way on the left, and house and sky in the upper left are all very badly recovered. This is due to occlusion.

As can be seen in Figure 6.25(d), the results show that only approximately 20% of the image is recovered with perfect depth, with another 15% recovered with only a single pixel of image error, giving a disappointing total image area of 35% recovered correctly (within tolerance). This surprising result was investigated. The remainder of this section shows the results of that investigation, and how it may assist in reducing errors in a real system.

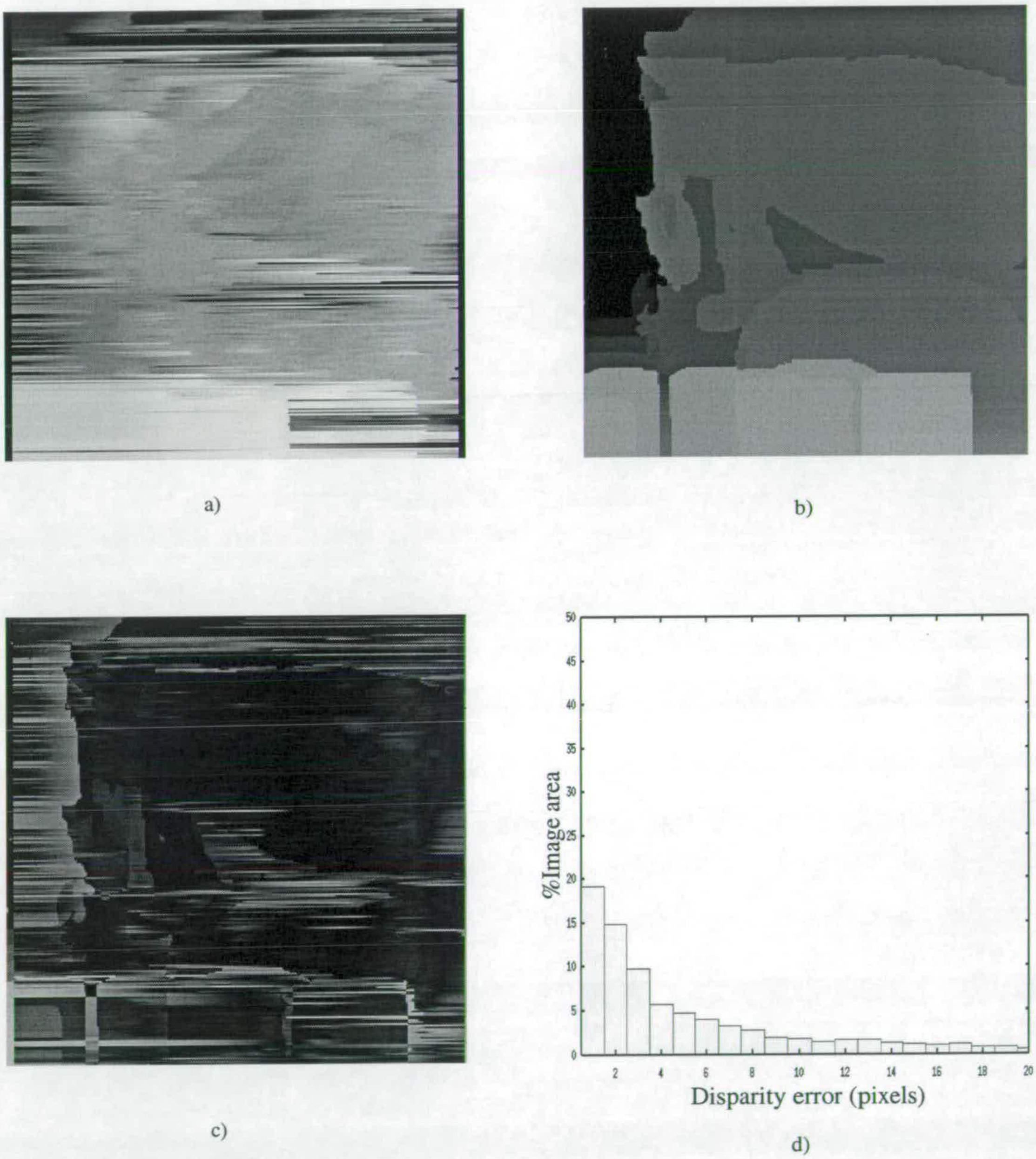


Figure 6.25: *An analysis of the error within this section 'cement' dataset. a) The disparity values. b) The estimated correct disparity values. c) The pixel difference between a. and b. d) The distribution of error within the image.*

6.4.2.1 Analysis of Results

In order to understand the errors, one must look at the differences between this dataset and the other two. As can be seen in Figure 6.24 much of the detail in the left image on the left hand side is omitted in the right image. This is due to the long baseline (1.5m), needed to recover the depth accurately. With such a long baseline, items in the foreground have large disparities.

Data Set	Smallest Disparity	Largest disparity	range
Corridor	1	12	12
Boxes	26	38	12
Cement	0	66	66

Table 6.4.2.1 shows the relative disparity ranges found in each of the data sets. The Corridor dataset has a short simulated baseline, and produces a very narrow range of disparities. The boxes dataset, also, despite having a larger baseline, is viewing a scene which is much more compressed along the 'z' depth axis, and hence produces a similar range of depths to the corridor. The cement dataset, however is viewing a scene with sky (nominally tagged at *infinity* depth, i.e. zero disparity), as well as barrels in the close foreground. This scene is also viewed with a large baseline. All of these produce a range of disparities five times larger than the other images.

The large difference in disparities between levels is advantageous, because it reduces the likelihood of objects to 'merge', as was seen in the boxes dataset, but increases the areas of occlusion.

To compound the problems, the camera lens also introduces a number of distortions which are not evident with small disparities, but which produce errors with larger disparities. This is because the difference in warping effect over a 10 to 20 pixel range is small, but over a 60 to 70 pixel range is very noticeable. This was compensated for using a small 'anti-fisheye' transform, and by using the top-left of the image frame (thereby keeping the largest disparities, the barrels, closer to the centre of the image frame). The reader may recall, that the assumption of spatial-based techniques is that the recording surface does not alter the image appearance at all. These transforms assist in asserting this assumption, but cannot compensate for them fully.

It was interesting to note that a large proportion of the error occurs at the more distant disparities. Given that the largest permissible error in the other two datasets was one disparity level, and given that the pixel depths were between 1 and 20 (approximately), it is asserted for the

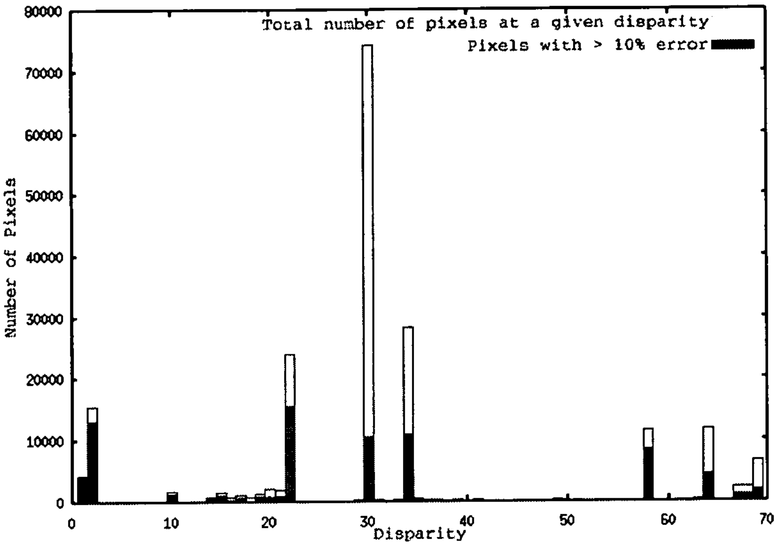


Figure 6.26: *The total number of pixels at each disparity level, and the amount of error at each disparity level within the image.*

following that 10% of error was permissible. Figure 6.26 shows the total number of pixels at each disparity level, and the calculated number of pixels at that disparity level with more than 10% of error.

Another way of looking at this data is to plot the amount of error as a percentage of the disparity level.

Figure 6.27 demonstrates that on average, the percentage of error increases with depth from the camera. In fact, due to the occlusions on the left, at distances greater than that of the shed behind the cement truck (i.e. disparity goes below 18), the average area approaches the value of the disparity itself. For example, at a disparity of 9 pixels (approximately 50m in this pair), the average error is equal to the disparity. Approaching infinite distance (the sky beyond, for example could be declared 'at infinity', as it has zero disparity at this resolution), the error becomes 10 times the disparity (1000% error). In this case, this error is exaggerated due to occlusion caused by the shed.

This can be seen in figure 6.28. The brighter values denote a higher percentage of error. The error is clipped at 100%, which is represented by white. The errors occurring in the distant portions of the image, as described above, can clearly be seen.

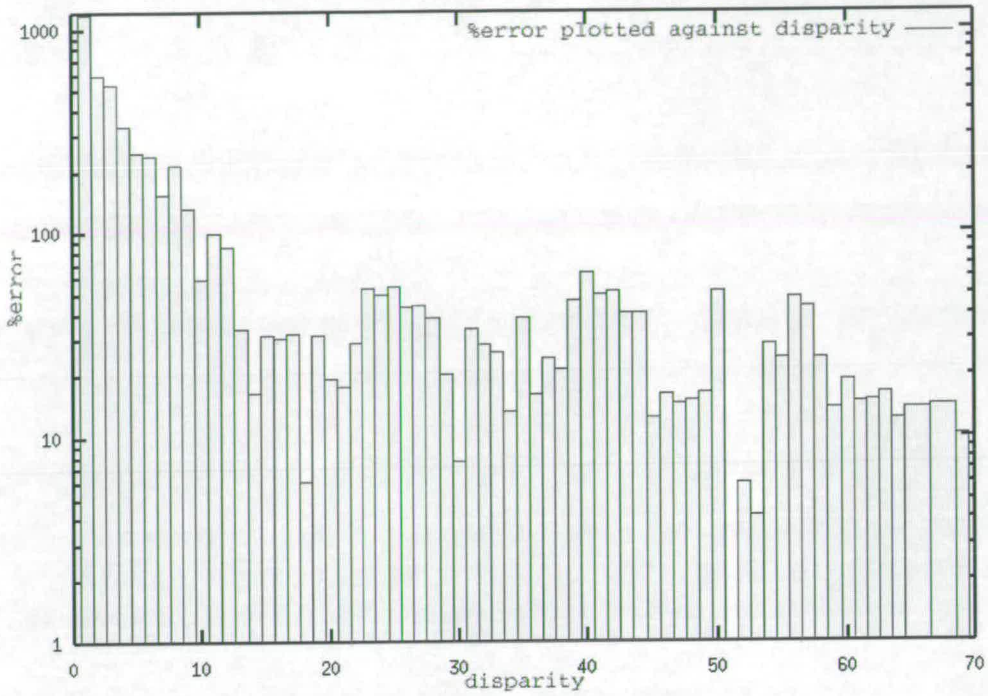


Figure 6.27: *The amount of error at each disparity level within the image, expressed as a percentage of the disparity level.*

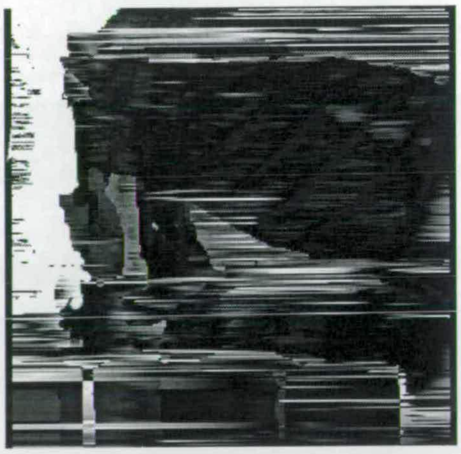


Figure 6.28: *The errors in the cement dataset plotted as percentage of disparity.*

This analysis may serve to assist in better understanding the nature of the errors, and why they occur within the images. It may also point to a way to speed up the matching process, and reduce error. Unfortunately, this kind of analysis assumes prior knowledge of the correct depth of each section of the image, which is impractical in a real application without another means of estimating depth (even if crude), and a 'memory' with which to intelligently manage those areas most likely to cause error. As can be seen, the errors are typically localised to the edges of correctly recovered depth objects, or at the edges of the image. They are also very horizontally-coherent, but with almost no vertical coherence. If it is possible to take a number of stereo images from slightly differing angles, these areas would stand out due to the lack of vertical coherence. That is to say that while varying the location of the cameras slightly would not affect the correctly matched areas significantly, those areas with errors would seem to have much more variation in depth, and would display little or no coherence in that variation, especially between adjacent epipolar lines. They are also identifiable by their particular horizontal "ramp" nature, as seen in the boxes dataset.

For example, if this were placed into a real-time system, and provided with continuous data, time-adjacent image frames could be used to keep track of the depths reported, form an approximation of depth (as an average of those reported), and therefore cleave all sections from the image whose estimated distances are deemed too large, or vary too much (i.e. unreliable areas).

6.5 Comparison to other Methods

It is difficult in Stereo Vision to compare the results to other systems. Each system uses different hardware, or uses proprietary scenes, chosen to best demonstrate the features of the system. For example, the output of this system seems very poor when compared against that of Kanade et al [71]. Of course, the computing power required, and the custom DSP boards used by their matcher, demonstrate that the SCP is complex problem to solve computationally, while maintaining a high-speed quality result.

Another system, by Dhond et al.[38], recovers depth in the presence of narrow occluding objects well. The runtimes are good, but once again, the computing power used is high, and there is no mention of how the system would perform in natural environments.

Only the work by Marr [36] or Cox [64], can be directly compared to this system, since they are similarly constructed, however, even these present limited results. Cox merely hints that the results point to pixel-based systems being 'contrary to popular belief, robust for a variety of images'. He also uses many more than two images to recover depth on particularly problematic images. He uses thirteen to recover an aerial photo of a castle. It seems that he was anxious to prove that his system was usable, but had to resort to many more than two images in some cases to prove the point. Other than the castle dataset, he uses only stereograms and computer generated faces or ellipsoids, i.e. very simple scenes.

Marr goes heavily into the algorithmic side of his theory (a basis for this project), and discusses in detail why the theory should work. Unfortunately, even by the time of his death in 1980, it seems he was unable to procure the computing power necessary to fully test all of his hypotheses. W.E.L. Grimson did much research based on Marr's hypotheses, but it seems he also favoured the frequential-based systems, and performed much more work on them. In 1984, he did some work with feature-based systems [37], as described earlier, but produced little qualitative result matter, and again based his experiments on random-dot stereograms.

There are very few *quantitative* reports on stereo vision. Most papers simply report a subjective success, discuss the algorithm, and skip the particulars of the scenes they have chosen. This has proven to be particularly vexing during this research.

However, I believe that pixel-based systems, given the current power of computing hardware, are equivalent to most other systems inspected, although perhaps not as robust, especially in the presence of severe occlusion or misaligned cameras. This will be discussed further in the Conclusions, in Chapter 7.

Chapter 7
Conclusions

7.1 Introduction

The solution to the SCP presented in this thesis has been drawn from a number of sources. As was shown in Chapter Two, it was hypothesised by Marr that such a solution is possible, and, given sufficient processing time, and good quality cameras, achievable. Grimson supported this with elementary systems run in limited environments. Cox, also supported this, along with other recent papers.

From the research done into other work in this field, it seems that few researchers are willing to delve into the exact problems faced by a system such as this. This is why Chapter Six, examines the particular problems in detail. Most SCP papers seem to highlight their successes, and use scenes specifically chosen to demonstrate their systems working (i.e. few difficult factors). The scenes in this project were chosen specifically to throw difficult problems at the matcher, to test its responses, and the results are, as expected, very telling.

Many *other* solutions to the SCP exist, and have been documented here. Unfortunately, testing stereo matching algorithms has not yet been standardised in the research community[24].

The wealth of research that exists in the field is staggering, and many of the same, re-cognizable names from the fields of biology, neurology, artificial intelligence, computer science, and electrical engineering occur time and again. One thing is clear, however, none of these researchers agree on what is the *correct* solution. Some, such as Marr, even contradict themselves in their various publications.

7.2 About the Method

This thesis set out to demonstrate that a system to solve the Stereo Correspondence Problem can be created using the technology of *Flow Networks*. The hypothesis was that using this technology for the disparity recovery engine could produce a system to solve a generalised form of the SCP, and also be optimisable such that it could be implemented on low-cost or general-purpose hardware.

The work presented in this thesis has shown that the use of flow networks does offer a viable solution to the SCP, at least in certain situations. It has also shown that this solution is straightforward to implement, using only standard algorithms from the areas of Vision, Artificial Intelligence, and Computer Graphics, and seems moderately robust to small quantities of noise. Despite the problems with occlusion, which are common to stereo systems, flow networks can be seen to be a good method for the production of dense depth maps.

There are some caveats to this assertion. Firstly, much of the optimisation discussed in this thesis pertains to parallel coplanar cameras. Secondly, if the number of image pixels with noise begins to grow large in proportion to those without, as shown in the additional noisy images in the corridor dataset, the pixel-skipping functionality of the flow networks cannot compensate. Lastly, regularly repeating patterns or textureless objects without any visible anchor cues contain too little differentiating information to allow a system such as this to match reliably. It has been suggested that linking a system such as this to an edge-matcher *could* assist in the anchoring of textureless areas. However doing so, would simply confuse areas of regularly repeating texture, since there would be many similar edges.

It has also shown that this type of solution should be capable of being implemented on general purpose hardware or embedded into a VLSI device. The network solver is very parallelisable, and the modest computational and memory requirements of the system, demonstrate that this is possible.

The last proposition in the hypothesis, taken from Marr's original list of requirements, was that such a system would have to be rapid.

Clearly readily available computing power is orders of magnitude greater than that which was available at the time of his death in 1980. Image capture devices are now so accurate and powerful that one can take photographs to a staggering resolution from orbiting satellites. So is the current state of computing and optics sufficient to make this method a rapid solution, according to Marr's requirements.

It is the author's opinion that the answer is "no". It will be a some time before such an approach will yield usable results in the natural environment. Taking an earlier hypothesis that a 400GHz machine would be needed to perform the matching in real time at 30 fps, then a machine of this power will be available in 15 to 20 years assuming that processor speeds continue to double every two years as at present'

The problem seems to stem mainly from a fundamental misunderstanding of the underlying data. The world, i.e. natural scenes, consist of high frequency data, and coherent objects often have regularly repeating, similarly coloured, textures. The world is also full of occlusions, parallax problems, and irregular shapes. The pixel-based stereo advocated by Marr and Poggio, originally, deconstructs these objects into individual pieces of data, thoroughly unconnected to each other. The network-reduction offered by Dykstra's algorithm can bind the pixels together to some degree, however when provided with such things as objects with repeating texture, a pixel-by-pixel agglomerative solution like this is faced with too many choices to make.

If the scene is not lit uniformly, then it can be seen that a pixel-by-pixel solution will be completely thrown off the correct path, even if attempts are made to gamma-correct the images before processing. Therefore spatial approaches will always work in limited environments.

This is not to say that the solution is *impossible.* Marr's original assumption, that given enough resources the solution is possible, is still correct. However, Alan Turing pointed such a thing out earlier, when he described his *Turing Machines*, and stated that "There was a duality between computing power and memory, given enough computing power, or time, anything is computable." The tractability of the SCP is merely a special case of this observation.

Unfortunately, time or computer memory are not limitless, and if a system such as this is to be used in real-world situations, it is clear that standard Von Neumann (single processor architecture) off-the-shelf computing hardware is still not fast enough to solve the SCP in real time. Only by resorting to parallel computing, or expensive DSP boards, can the process be accelerated.

7.3 About the field

It seems, therefore, that if any solution is going to win, it must be more relaxed in its attitude towards data integrity. By this, it is meant that lighting can be non-uniform, objects can be occluded or reflective, and the cameras can become misaligned. A purely pixel-based solution, probably cannot easily be as flexible as is required for this to become realisable. There will always need to be some other comparative process to (figuratively speaking) "keep an eye" on the matcher. In the early part of this thesis, it was stated that the purpose of this may be to assist a human operator, and therefore the human would be the comparative process.

Frequential methods, while computationally expensive, are the most likely candidate, of the purely passive methods. Since frequency data is not affected by the intensity values of the spatial data, merely its orientation and position, this is good from an imperfect lighting perspective. Area-based methods are also promising due to their ability to recognise chunks of objects contextually, and are not easily fooled by small amounts of impulsive noise, or occlusion. Unfortunately area based methods suffer their own brands of problems, as described earlier. So, at the moment, there does not seem to *be* any right or wrong solution. The subject of the brain has been discussed earlier, but exactly *how* can it achieve such accurate results?

Marr's opinion, and also mine is that the human brain can perform so well because it has a wealth of experience to draw on, and a few tricks, thanks to the eyes. However, he also believed that the brain's prior experience was not *necessary*.

All through the thesis, the problem of obtaining correlation data to test hypotheses has been discussed. Where does the brain get its correlation information? In later life, after much experience, objects can be more easily identified even before the distance measurement step. An easy test for this is to stare at something, say this thesis, and identify objects in your peripheral vision. These items are not being measured for distance. It is easy to prove that humans perform no distance measurement far from the centre of vision. If a bird, or ball flies through the periphery, even at some distance, it can cause the person to flinch, or duck, and turn towards the object because the person has no idea of its distance. One can also cover an eye, and look at something, and still guess depth, based upon knowledge of the object's size. Ames' Room is good example of a system, which can fool this prior-knowledge architecture.

How do humans therefore learn distance measurement? There must be some learning stage so that when we do reach visual maturity we can indeed identify objects. The eyes play an important part of this. The eyes, unlike the cameras used in this project, are not static. The brain can control them, and can request them to fixate on any point ahead of the face. The brain can therefore ask the eyes to look along an angle, and with a convergence of some amount. The visual cortex can then compare the images falling onto the centres of the retinas and report to the brain whether it is, indeed fixating on an object. Since the eyes can only (normally) converge, there is a limited range of angles that the brain can request, and therefore, using any simple feedback mechanism can "home in" on the correct convergence, and test its hypothesis.

So, how does the brain know that a distance x corresponds to a vergence angle θ? Very young babies have difficulty focusing on objects. As motor skills increase, they can, through a gradual process of experimentation, co-ordinate limbs, to grasp objects, and once the object is grasped, look at it, and experiment with it, say by trying to insert it into the mouth (during the oral stage of development), or throw it away in a desired direction.

The human visual system likely begins largely untrained, and gradually achieves a level of competency whereby it can make hypotheses and test them passively, by refocusing the eyes, or actively by touching.

Computers have no such system, and must therefore solve the SCP without help from intelligence. Many examples have been given so far on other shape-from-x solutions. Many assume structured lighting or conditions, but give great hope to this researcher that they can be incorporated into later systems to give some 'intelligence' to a depth matcher, in a similar way to real system.

Work is currently being carried out in the field of active-mounting cameras, and convergeable recording surfaces. While the various researchers argue philosophically about which system has more evidence to support its physiological validity, it is this researcher's opinion that the ability for the system, based on some mix of frequential and area-based matching, to actually test its hypotheses, based on prior knowledge, using steerable cameras which will give a truly flexible system which will operate in almost any environment.

7.4 Future Research

The stereo matcher given in this thesis is a good starting point to a pixel-based solution. While the results are comparable to other systems, its flexibility has allowed many experiments to be performed on the variables, and weightings. The object-based approach to its design has also allowed several different network reduction mechanisms to be employed. These all allow a system such as this to be expanded and improved in future.

7.4.1 Occlusion Detection

It is the author's opinion that the greatest benefit (as highlighted in Chapter 6) to a pixel-based solution, would be to incorporate some knowledge of occlusion. By this, it is suggested that the system be allowed to hypothesise that an area which is displaying the "ramp" nature as shown in the boxes dataset in Chapter 6, be marked as a possible occlusion, and treated accordingly. How this is performed is a topic for future work. The work by Dhond[38], is a good example of a system which can overcome (at least a small amount of) occlusion.

The work by Cox and Fielding[25,64] goes the furthest in identifying and removing occlusion, and shows that detecting when a match should be surrendered in favour of an occlusion is not only possible, but also highly workable. Even they, however, still cannot eradicate all of the issues with occlusion.

7.4.2 Temporally Adjacent Image Frames

The system given in this thesis was presented with static images. As has been mentioned before, temporally adjacent image frames could help to drastically reduce the number of errors, and could help to track recovery problems, and/or optimise the matching process.

7.4.3 Textureless Area Handling

Another major problem with this system was the recovery of depth of objects of featurelessness, especially whose boundaries extend out-with the image frame, or are occluded in part by other objects. Much work was put into solving this problem, as can be seen within the Thesis, however no satisfactory solution was arrived at. It is interesting to note, perhaps, that in none of the texts researched, was this problem highlighted. Only the problems of repetitive, and very high frequency texture were investigated. Other researchers may have come across this, and chosen to ignore it; have come across it and decided it is not a significant problem; have come across it, and decided to hide their own systems' inability to cope with it, or have not come across it at all?

7.4.4 Optimisation

As with all systems designed for open-access, and ease of modification, there are many points where optimisation could be employed. The block- removal system, for example, proved to be a great optimisation tool for simple images, however it slowed the system down on extremely complex natural data. It is clear that for a pixel-based solution, much optimisation of the network-reduction step is required.

7.5 Concluding Remarks

Some promising results have been obtained, which suggest that a pixel-based stereo matcher based upon the flow networks model could provide a viable solution. Many avenues could not be explored, due to time and resource constraints.

However, the results suggest that there are too many points of failure in a pixel-based stereo solver. While this project has confirmed that stereo *can* be recovered in a wide range of scenes using this process and that using available computing and imaging hardware is sufficient for crude results; it is clear that there are simply too many data-dependent issues to be solved *solely* at such a low level.

It is therefore the conclusion of this thesis, that after much research and intricate investigation of the particular issues, which have been individually highlighted, pixel-based stereo on its own cannot generate accurate dense depth maps for consumer use. This conclusion is likely to be true until such time as:

- Computing power is many hundreds of times more powerful than it is now,
- ASIC cost drops to the point where hundreds of very high-performance parallel DSPs can be placed inside consumer electronics,
- The cameras can always be guaranteed to be perfectly aligned, clean and exhibit well-understood distortion.
- A formulation of generalised "common sense", or human-style learned knowledge can be supplanted into a stereo system.

As has been found by many researchers in the papers listed, it is unfortunate that despite the promising initial results, there are simply too many points of failure which require recalibration, extra time, or manual intervention, to permit such a solution on its own, to be useful in the natural environment.

References

[1] D. Marr and T. Poggio, "A theory of human stereo vision," in *Proceedings of the Royal Society of London*, vol. B204, 1979.

[2] R. C. Gonzalez and R. E. Woods, *Digital Image Processing.* Student Series, Addison Wesley, Sep 1993.

[3] M. Okutomi and T. Kanade, "A multiple-baseline stereo," Internal Report CMU-CS-90-189, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, November 1990.

[4] U. R. Dhond and J. Aggarwal, "Structure from stereo - a review," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 19, pp. 1489–1510, Dec. 1989. Cool Paper.

[5] T. M. B. K. P. Han and Y. H. Ha., "Hybrid stereo matching with a new relaxation scheme of preserving disparity discontinuity," *Pattern Recognition*, vol. 33, pp. 767–785, 2000.

[6] K. P. Han, K. W. Song, E. Y. chung, S. Cho, and Y. H. Ha, "Stereo matching using genetic algorithm with adaptive chromosomes," *Pattern Recognition*, vol. 34, pp. 1729–1740, September 2001.

[7] C. V. Jawahar and P. Narayanan, "Generalised correlation for multi-feature correspondence," *Pattern Recognition*, vol. 35, pp. 1303–1313, June 2002.

[8] D. Scharstein and R. Szeliski, "Stereo matching with non-linear diffusion," in *Proceedings IEEE 1996 computer Society Conference on Computer Vision and Pattern Recognition*, pp. 343–350, IEEE, 1996.

[9] C. H. Huang and J. H. Wang, "Stereo correspondence using hopfield networks with multiple constraints," in *IEEE International Conference on Systems, Man, and Cybernetics*, vol. 2, pp. 1518–1523, 2000.

[10] N. DiRuocco, A. Vitale, and S. Vitulano, "Artificial intelligence in vision," in *Proceedings of the 11th IAPA Internation Conference on Pattern Recognition*, vol. 1 (Conference A: Computer Vision and Applications), pp. 453–456, 1992.

[11] N. Qian and Y. Zhu, "Physiological computation of binocular disparity," *Vision Research*, vol. 37, pp. 1811–1827, July 1997.

[12] M. Zaki, A. El-Ramsisi, and R. Omran, "A soft computing approach for recognition of occluded shapes," *The Journal of Systems and Software*, vol. 51, pp. 73–83, 2000.

[13] Y. Ruichek, J. G. Postaire, and J. L. Bruyelle, "A neural approach for obstacle detection with a linear stereoscopic sensor," *Mathematical and Computer Modelling*, vol. 27, pp. 215–228, May 1998.

[14] J. Chai and S. De-Ma, "Robust epipolar geometry estimation using genetic algorithm," *Pattern Recognition Letters*, vol. 19, pp. 829–838, July 1998.

[15] D. V. Papadimitriou and T. J. Dennis, "A stereo disparity algorithm for 3d model construction," in *Proceedings of the 5th International Conference on Image Processing Applications*, no. 410 in -, pp. 178–182, July 1995.

[16] R. Chung and S.-K. Wong, "Stereo calibration from correspondences of otv projections," *IEE. Proceedings of Vision, Image + Signal Processing*, vol. 142, pp. 289–296, October 1995. OTV.

[17] Y. Kanazawa and K. Kanatani, "Reliability of 3-d reconstruction by stereo vision," *IEICE Transactions on Information and Systems*, vol. E78-D, pp. 1301–1306, Oct 1995.

[18] H. Sahabi and A. Basu, "Analysis of error in depth perception with vergence and spacially varying sensing," *Computer Vision and Image Understanding*, vol. 63, pp. 447–461, May 1996.

[19] S. B. Marapane and M. M. Trivedi, "Region-based stereo analysis for robotic applications," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 19, pp. 1447–1463, November 1989.

[20] D. N. Bhat and S. K. Nayar, "Stereo in the presence of specular reflection," in *Proceedings of the Fifth International Conference on Computer Vision*, pp. 1086–1092, June 1995.

[21] L.-H. Chen and W.-C. Lin, "Visual surface segmentation from stereo," *Image and Vision Computing*, vol. 15, pp. 95–106, Feb 1997.

[22] Y. Suya, L. Jian, and W. Faguan, "A novel stereo technique based on anisotropic character of human visual system," in *Proceedings of TENCON '93, the IEEE Region 10 Conference on Computer, Communications, Control and Power Engineering.*, vol. 4, pp. 264–266, IEEE, 1993.

[23] D. V. Papadimitriou and T. J. Dennis, "Surface triangulation for 3-d scene model construction from stereo," *IEE Proceedings Vision Image Signal Processing*, vol. 143, pp. 310–314, October 1996. calibrated stereo model based.

[24] D. Craievich, B. Barnett, and A. C. Bovik, "A stereo visual pattern image coding system," *Image and Vision Computing*, vol. 18, pp. 21–37, 1999.

[25] G. Fielding and M. Kam, "Weighted matchings for dense stereo correspondence," *Pattern Recognition*, vol. 33, pp. 1511–1524, 2000.

[26] A. Mayoral and M. J. Perez-Ilzarbe, "Competitive hopfield neural network for stereo vision correspondence," in *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks*, vol. 5, pp. 464–469, 2000.

[27] A. Y.-K. Ho and T.-C. Pong, "Cooperative fusion of stereo and motion," *Pattern Recognition*, vol. 29, no. 1, pp. 121–130, 1996.

[28] B. Arion, N. Yang, and F. Devos, "A novel stereovision architecture for real-time obstacle detection," in *IEEE First International Conference on Algorithm and Architecture for Parallel Processing*, vol. 1, pp. 395–403, IEEE, 1995.

[29] A. D. Jepson and M. R. M. Jenkin, "The fast computation of disparity from phase differences," in *Proceedings of CVPR '89: The IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 389–403, IEEE, 1989.

[30] B. Ross, "A practical stereo vision system," in *Proceedings 1993 IEEE Computer Society Conference on Computer Vision and PAttern Recognition*, pp. 148–153, IEE Computer Society Press, 1993.

[31] A. K. Jain and R. Hoffman, "Evidence-based recognition of 3-d objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, pp. 783–801, November 1988. more randomness.

[32] J. M. Brady, "Problems of robotics," Tech. Rep. OUEL 1746/88, Univeristy of Oxford, Dept. Eng Sci, Oxford University, Parks Road, Oxford, OX1 3PJ U.K., 1988.

[33] H. Yuhara, A. Terauchi, M. Saka, M. Habaguch, and M. Kawai, "Stereo vision sensor," *Society for Automotive Engineers of Japan (JSAE) Review*, vol. 21, pp. 529–534, 2000.

[34] D. H. Ballard and C. M. Brown, *Computer Vision*. ISBN 0-13-1653316-4, Prentice / Hall, 1982.

[35] M. Adjouadi, F. Candocia, and J. Riley, "Exploiting walsh-based attributes to stereo vision," *IEEE Transactions on Signal Processing*, vol. 44, pp. 409–420, Feb 1996.

[36] D. Marr and T. Poggio, "Cooperative computation of stereo disparity," *Science*, vol. 194, pp. 283–287, October 1976.

[37] W. E. L. Grimson, "Computational experiments with a feature based stereo algorithm," A.I. Memo 762, Massachusetts Institute of Technology, January 1984.

[38] U. R. Dhond and J. Aggarwal, "Stereo matching in the presence of narrow occluding objects using disparity search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, pp. 719–724, July 1995.

[39] D. C. Brockelbank and Y. H. Yang, "An experimental investigation in the use of color in computational stereopsis," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 19, pp. 1365–1383, November 1989.

[40] E. H. Adelson and J. Y. Wang, "Single lens stereo with a plenoptic camera," *PAMI*, vol. 14, pp. 99–106, Feb 1992.

[41] P. Lavoie and D. Ionescu, "Automatic generation of epipolar images," in *1994 Canadian Conference on Electrical and Computer Engineers*, vol. 2, pp. 588–591, IEE, 1994.

[42] S. Panis, M. Zeigler, and J. Cosmas, "System approach to disparity estimation," *Electronics Letters*, vol. 31, pp. 871–873, May 1995.

[43] A. Rojas, A. Calvo, and J. Munoz, "A dense disparity map of stereo images," *Pattern Recognition Letters*, no. 18, pp. 385–393, 1997.

[44] R. Ginhoux and J. S. Gutmann, "Model-based object tracking using stereo vision," in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1226–1232, 2001.

[45] A. Zanela and S. Teraglio, "Sensing the third dimension in stereo vision systems: a cellular neural networks approach," *Engineering Applications of Artificial Intelligence*, no. 11, pp. 203–213, 1998.

[46] Y. Do, "Application of neural networks for stereo-camera calibration," *IJCNN '99. International Joint Conference on Neural Networks*, vol. 4, pp. 2719–2722, 1999.

[47] A. Khamene and S. Negahdaripour, "Building 3d elevation maps of sea-floor scenes from underwater stereo images," in *OCEANS '99 MTS/IEEE. Riding the Crest into the 21st Century*, vol. 1, pp. 64–70, September 1999.

[48] E. Izquierdo and V. Guerra, "Improving efficiency of linear techniques to estimate epipolar geometry," *Electronics Letters*, vol. 37, pp. 952–954, July 2001.

[49] M. Hariyama, T. Takeuchi, and M. Kameyama, "Reliable stereo matching for highly-safe intelligent vehicles and its vlsi implementation," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, vol. IV, pp. 128–133, 2000.

[50] M. Hariyama, T. Takeuchi, and M. Kameyama, "Vlsi processor for reliable stereo matching based on adaptive window-size selection," in *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, (Seoul, Korea), pp. 1168–1173, May 2001.

[51] P. Moallem, K. Faez, and J. Haddadnia, "Reduction of the search space region in the edge based stereo correspondence," in *ICIP01*, pp. II: 149–152, 2001.

[52] R. K. K. Yip and W. P. Ho, "A multi-level dynamic programming method for stereo line matching," *Pattern Recognition Letters*, vol. 19, pp. 839–855, 1998.

[53] C. Tomasi and T. Kanade, "Shape and motion without depth," Internal Report CMU-CS-90-128, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, May 1990.

[54] T. Poggio, "A theory of how the brain might work," Internal Report A.I. Memo No. 1253, Massachusetts Institute of Technology, December 1990.

[55] K. Yamamoto, "Optimization approaches to constraint satisfaction problems in computer vision," *Image and Vision Computing*, vol. 13, pp. 335–340, Jun 1995.

[56] M. Okutomi and Y. Katayama, "A simple stereo algorith, to recover precise object boundaries and smooth surfaces," in *Proceedings of the IEEE Workshop on Stereo and Multi-Baseline Vision (SMBVi01)*, vol. 2, pp. 158–165, 2001.

[57] A. Busboom and R. Schalkoff, "Active stereo vision and direct surface parameter estimation: Curve-to curve image plane mappings," in *IEE Proceedings Vision Image Signal Processing*, vol. 143, pp. 109–117, April 1996.

[58] R. Horaud and T. Skordas, "Stereo correspondence through feature grouping and maximal cliques," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, pp. 1168–1180, Nov 1989.

[59] K. boyer and A. Kak, "Structural stereopsis for 3-d vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, pp. 144–166, March 1988.

[60] J. Zhou and B. Ghosh, "A gradient algorithm for stereo matching without correspondence," *IEEE Transactions on Automatic Control*, vol. 41, pp. 1671–1676, Nov 1996.

[61] D. Marr, G. Palm, and T. Poggio, "Analysis of a cooperative stereo algorithm," A.I. Memo 446, Massachusetts Institute of Technology, October 1977.

[62] D. Kosmopoulos and T. Varvarigou, "Automated inspection of gaps on the automobile production line through stereo vision and specular reflection," *Computers in Industry*, vol. 46, pp. 49–63, August 2001.

[63] E. L. Walker, "Knowledge-based image understanding using incomplete and generic models," in *Proceedings of the 1993 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 694–700, IEEE, 1993.

[64] I. J. Cox, S. L. Hingorani, and S. B. Rao, "A maximum likelihood stereo algorithm," *Computer Vision and Image Understanding*, vol. 63, pp. 542–567, May 1996.

[65] D. Marr, *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. W.H. Freeman and Co., 1982.

[66] M. Brady, *The simulation of Human Intelligence*, ch. 3, Computational Vision, pp. 121–150. Blackwell, 1993. ISBM 0-631-18587-9.

[67] R. Koch, "3-d surface reconstruction from stereoscopic image sequences," in *Proceedings Fifth International Conference on Computer Vision*, pp. 109–114, IEE Computer Society Press, Jun 1995.

[68] D. N. Bhat and S. K. Nayar, "Ordinal measures for visual correspondence," in *Proceedings IEEE 1996 Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 351–357, IEEE, June 1996.

[69] E. Grosso, G. Sandini, and M. Tistarelli, "3-d object reconstruction using stereo and motion," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 19, pp. 1465–1472, November 1989.

[70] T. Kanade and M. Okutomi, "A stereo matching algorithm with an adaptive window: Theory and experiment," Internal CS Report CMU-CS-90-120, School of Computer Science, Carnegie Mellon University, Pittsburghg, PA, 15213, April 1990.

[71] T. Kanade, A. Yoshida, K. Oda, H. Kano, and M. Tanaka, "A stereo machine for video-rate dense depth mapping and its new applications," in *Proceedings IEEE 1996 Conference for Computer Vision and Pattern Recognition*, vol. 17 No.7, pp. 719–724, IEEE, Jul 1996.

[72] S. Pankanti and A. K. Jain, "Integrating vision modules: Stereo, shading, grouping, and line labeling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, pp. 831–842, Sep 1995.

[73] U. Franke and A. Joos, "Real-time stereo vision for urban traffic scene understanding," in *Proceedings of the IEEE Intelligent Vehicles Symposium 2000*, pp. 273–279, October 2000.

[74] J. J. Kweon, D. K. Kang, and S. D. Kim, "A stereo matching algorithm using line segment features," in *Coference Proceedings of TENCON '89: 4th IEEE Region 10 International Conference on Information Technologies for the '90's (E/Sup, 2/C/Sup, Energy, Electronics, Computers and Communications)*, pp. 589–592, IEEE, 1989.

[75] P. L. Rosin, "Augmenting corner descriptors," *Graphical Models and Image Processing*, vol. 58, pp. 286–294, May 1996.

[76] G. Erten and F. M. Salam, "Real time realization of early visual perception," *Computers and Electrical Engineering*, vol. 25, pp. 379–407, 1999.

[77] M. L. Minsky, "Estimating stereo disparities," Tech. Rep. Vision Memo. No. 121, Massachusetts Institute of Technology, February 1967.

[78] J. Neubert, T. Hammond, N. Guse, Y. Do, Y. Hu, and N. Ferrier, "Automatic training of a neural net for active stereo 3d reconstruction," in *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, (Seoul, Korea), pp. 2140–2146, May 2001.

[79] S. Dickson, B. thomas, W. P. Mackeown, and P. Goddard, "Computer vision applied to the detection and localisation of accoustic neuromas from head mp images," *IEE Proccedings Vision Image Signal Processing*, vol. 143, pp. 265–272, August 1996. MEdical Stuff with edges stuff.

[80] H. A. Simon, "Lessons from perception for chess-playing programs (and vice versa)." CMU - Computer Science Research Review 1972-1973 Annual Report, Carnegie Mellon University, September 1973.

[81] M. Nishigaki, M. Saka, T. Aoki, H. Yuhara, and M. Kawai, "Fail output algorithm of vision sensing," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, vol. IV, pp. 581–584, 2000.

[82] W. vanderMark and F. C. A. Groen, "Stereo based navigation in unstructured environments," in *IEEE Instrumentation and Measurement Technology Conference*, (Budapest, Hungary), pp. 2038–2043, IEEE, May 2001.

[83] T. Dean, J. Allen, and Y. Aloimonos, *Artificial Intelligence, Theory and Practice*. Benjamin / Cummings, 1995. ISBN 0-8053-2547-6.

[84] C. M. Brown, "Issues in selective perception," *IEEE*, pp. 21–30, 1992.

[85] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organisation in the brain," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[86] M. Minsky and S. Papert, *Perceptrons: An introduction to Computational Geometry*. MIT Press, 1969.

[87] R. Penrose, *The simulation of Human Intelligence*, ch. 1, Setting the Scene: The Claim and the Issues., pp. 1–32. Blackwell, 1993. ISBN 0-631-18587-9.

[88] M. Minsky, "Stereo and perspective calculations," Artificial Intelligence (Project MAC) Report AI Memo. No. 143, Massachusetts Institute of Technology, September 1967.

[89] T. Poggio, H. K. Nishihara, and K. R. K. Nielsen, "Zero-crossings and spaciotemporal interpolation in vision: aliasing and electrical coupling between sensors," A.I. Memo 675, Massachusetts Institute of Technology, May 1982.

[90] S. Ullman, "Visual routines," A.I. Memo 723, Massachusetts Institute of Technology, June 1983.

[91] C. E. Shannon and W. Weaver, *The Mathematical Theory of Communication*. University of Illinois Press, Illinois., 1949.

[92] C. Koch, J. Marroquin, and A. Yuille, "Analog 'neuronal' networks in early vision," Internal Memo A.I. Memo 751, Massachusetts Institute of Technology AI Laboratory, June 1985.

[93] I. Beiderman, "Recognition by components: A theory of human image understanding," *Psychological Review*, vol. 94, no. 2, pp. 115–147, 1987.

[94] L. Stark and K. Bowyer, "Achieving generalised object recognition through reasoning about associations of function to structure," *IEEE Transactions on Pattern and MAchine Intelligence*, vol. 13, Oct. 1991.

[95] M. M. Trivedi and A. Rosenfeld, "On making computers 'see'," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 19, pp. 1333–1334, November 1989. editorial.

[96] S. Hongo, N. Sonehara, and I. Yoroizawa, "Edge-based binocular stereopsis algorithm - a matching mechanism with probabilistic feedback," *Neural Networks*, vol. 9, no. 3, pp. 379–395, 1996.

[97] W. E. L. Grimson, "Binocular shading and visual surface reconstruction," A.I. Memo 697, Massachusetts Institute of Technology, August 1982.

[98] D. Marr, E. Hildreth, and T. Poggio, "Evidence for a fifth, smaller channel in early human vision," A.I. Memo 541, Massachusetts Institute of Technology, August 1979.

[99] J. Weng, "Signal reconstruction from windowed fourier phase," in *IEEE conference on Accoustics, Speech and Signal Processing* (IEEE, ed.), vol. 4, pp. 145–148, IEEE, March 1992.

[100] U. Buker, S. Drue, N. Gotze, G. Hartman, B. Kalkreuter, R. Stemmer, and R. Trapp, "Vision-based control of an autonomous disassembly station," *Robotics and Autonomous Systems*, vol. 35, pp. 179–189, 2001.

[101] P. Fua and Y. Leclerc, "Taking advantage of image-based and geometry-based constraints to recover 3-d surfaces," *Computer Vision and Image Understanding*, vol. 64, pp. 111–127, Jul 1996.

[102] Y. T. Zhou and R. Chellappa, "Neural network algorithms for motion stereo," in *Proceedings of IJCNN: International Joint Conference on Neural Networks*, vol. 2, pp. 251–258, IEEE TAB Neural Network Committee, 1991.

[103] D. McCartney, D.E.Sheat, G. Chamberlin, and D. Travis, "Telecommunications applications for 3-d imaging systems," in *Proceedings SPIE - The International Society for Optical Engineering*, vol. StereoScopic Displays and Applications IV, (San Jose, California), pp. 29–35, SPIE, Feb 1993.

[104] J. Hsu, Z. Pizlo, D. M. Chelberg, C. F. Babbs, and E. J. Delp, "Issues in the design of studies to test the effectiveness of stereo imaging," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 26, pp. 810–819, Nov 1996.

[105] J. F. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, pp. 679–698, January 1985.

[106] D. Marr and E. Hildreth, "Theory of edge detection," *Proceedings of the Royal Society of London*, vol. B207, pp. 187–217, 1980.

[107] S. Chen, Z. Kazi, R. Foulds, and D. Chester, "Color and three-dimensional vision-based assistive telemanipulation," *Image and Vision Computing*, vol. 16, pp. 265–274, 1998.

[108] K. Kato, H. Ishiguro, and S. Tsuji, "Estimating precise edge position by camera motion," *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 824–829, Oct 1996.

[109] C. Dorai and A. Jain, "Cosmos-a representation scheme for free-form surfaces," *Transactions of the IEEE*, August 1995.

[110] T. Moons, L. V. Gool, M. Proesmans, and E. Pauwels, "Affine reconstruction from perspective image pairs with a relative object-camera translation in between," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, pp. 77–82, Jan 1996.

[111] U. R. Dhond and J. Aggarwal, "A cost benefit analysis of a third camera for stereo correspondence," *International Journal of Computer Vision*, vol. 6, no. 1, pp. 39–58, 1991.

[112] S. Sakamoto, I. J. Cox, and J. Tajima, "A multiple-baseline stereo for precise human face acquisition," *Pattern Recognition Letters*, vol. 18, pp. 923–931, 1997.

[113] C. Chang and S. Chatterjee, "Depth from stereo image flow," in *Conference Proceedings of the 1989 IEEE International Conference on System, Man and Cybernetics*, vol. 2, pp. 586–591, IEEE, 1989.

[114] J. Ruiz-Alzola, C. Alberola-Lopez, and J.-R. C. Corredera, "Model-based stereo-visual tracking: Covariance analysis and tracking systems," *Signal Processing*, vol. 80, pp. 23–43, 2000.

[115] Y. Ninomiya, S. Matsuda, M. Ohta, and Y. Harata, "A real-time vision for intelligent vehicles," in *Proceedings of the Intelligent Vehicles '96 Symposium*, pp. 315–320, IEEE, 1996.

[116] N. Molton, S. Se, M. Brady, D. Lee, and P. Probert, "Robotic sensing for the partially sighted," *Robotics and Autonomous Systems*, vol. 26, pp. 185–201, 1999.

[117] R. J. Boys, "Frequently asked questions of comp.arch.bus.vmebus," FAQ 9, San Jose, California, http://www.ee.ualberta.ca/archive/vmefaq.html, January 1996.

[118] P. I. Corke, P. A. Dunn, and J. E. Banks, "Frame-rate stereopsis using non-parametric transforms and programmable logic," in *Proceedings of the 1999 IEEE International Conference on Robotics and Automation*, vol. 3, (Detroit, Michigan, USA), pp. 1928–1933, May 1999.

[119] E. B. Gamble, D. Geiger, T. Poggio, and D. Weinshall, "Integration of vision modules and labelling of surface discontinuities," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 19, pp. 1576–1580, Nov. 1989.

[120] D. J. Kriegman, E. Treindl, and T. O. Binford, "Stereo vision and navigation in buildings for mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 5, pp. 792–803, Dec 1989. Epipolar Line justification.

[121] W. E. L. Grimson and T. Lozano-Perez, "Model-based recognition and localization from sparse range or tactile data," A.I. memo 738, Massachusetts Institute of Technology, August 1983.

[122] M.-J. Chen, L.-G. Chen, C.-N. Cheng, and M. Chen, "Efficient hybrid tree/linear array architectures for block-matching motion estimation algorithms," *IEE Proceedings Vision Image Signal Processing*, vol. 143, pp. 217–222, August 1996. Discussion of the crapness of Block-Matching.

[123] A. Blake and A. Zisserman, "Using weak continuity constraints - what they are, and how they do it.." Internal Booklet, May 1985.

[124] A. Blake and A. Zisserman, "Using weak continuity constraints - what they are and how they do it.," *Dept. computer Science, Edinburgh University internal Memo*, 1990. useless paper by a useless wanker.

[125] C. Koch and T. Poggio, "Biophysics of computation: Neurons, synapses and membranes," *A.I. Memo 795*, October 1984.

[126] Y. S. Chen, Y. P. Hung, and C. S. Fuh., "Fast block matching algorithm based on the winner-update strategy," *IEEE Transactions on Image Processing*, vol. 10, pp. 1212–1222, August 2001.

[127] Y. F. Wan, F. Cabestaing, and J.-C. Burie, "A new edge detector for obstacle detection with a linear stereo vision system," in *Proceedings of the Intelligent Vehicles '95 Symposium*, pp. 130–135, IEEE, 1995.

[128] E. C. Hildreth, "Implementation of a theory of edge detection," msc. thesis and report no. ai-tr-579, Massachusetts Institute of Technology, AI Laboratory, April 1980.

[129] D. Marr and E. Hildreth, "Theory of edge detection," A.I. Memo 518, Massachusetts Institute of Technology, April 1979.

[130] J. F. Canny, "Finding edges and lines in images," in *MSc. Thesis and Tech Report No. 720*, Massachusetts Institute of Technology, AI Laboratory, June 1983.

[131] A. C. Guyton, *Textbook of Medical Physiology*. Philadelphia, PA 19106: W.B. Saunders Co., eigth edition ed., 1991.

[132] R. W. Conners and C. T. Ng, "Developing a quantitative model of human preattentive vision," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 19, pp. 1384–1394, November 1989.

[133] O. Laligant, F. Truchetet, and J. Miteran, "Edge detection by multiscale merging," in *Proceedings IEEE-SP International Symposium on Time-Frequency and Time-Scale Analysis*, pp. 237–240, IEEE, Oct 1994.

[134] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. Cambridge, Massachusetts: The Massachusetts Institute of Technology Press, 11 ed., 1990.

[135] C. Sun, "Rectangular subregioning and 3-d maximum-surface techniques for fast stereo matching," *IEEE Workshop on Stereo and Multi-Baseline Vision (in conjunction with CVPR'01)*, pp. 44–53, December 2001.

[136] B. Julesz, "Binocular depth perception of computer-generated patterns," *Bell Systems Technical Journal*, vol. 39, pp. 1125–1162, 1960.

[137] B. Julesz, *Foundations of Cyclopean Perception*. Chicago: University of Chicago Press, 1971.

[138] D. Scharstein, "Stereo vision for view synthesis," in *Proceedsings of IEEE 1996 Computer society Conference on Computer Vision and PAttern Recognition*, pp. 852–858, IEEE, 1996.

[139] D. V. Papadimitriou and T. J. Dennis, "Stereo disparity analysis using phase correlation," *ELETTERS*, Jul 1994.

[140] C. L. Pagliari and T. J. Dennis, "Disparity estimation using edge-oriented classification in the dct domain," *Electronics Letters*, vol. 34, June 1998.

[141] W. Grimson, "A computer implementation of a theory of human stereo vision," *Massachusetts Institute of Technology Internal Report A.I. memo No. 565*, pp. 1–59, January 1980.

[142] H. K. Nishihara, "Prism: A practical real-time imaging stereo matcher," A.I. Memo 780, Massachusetts Institute of Technology, May 1984.

[143] S. Pollard, J. Mayhew, and J. Frisby, "Pmf: A stereo correspondence algorithm using a disparity gradient limit.," in *Perception*, vol. 14, pp. 449–470, ,, 1985.

[144] K. Shanmukh and Y. V. Venkatesh, "Generalised scheme for optimal learning in recurrent neural networks," *IEE Proceedings Vision Image Signal Processing*, vol. 142, pp. 71–77, April 1995. Improved Learning for N's and discussion of NN's.

[145] S. Nichani, "Solving the correspondence problem using a hopfield network," in *IEEE International Conference on Neural Networks*, vol. 6, pp. 4107–4112, IEEE, 1994.

[146] Y. V. Venkatesh and N. Rishikesh, "Self-prganizing neural networks based on spatial isomorphism for active contour modelling," *Pattern Recognition*, vol. 33, pp. 1239–1250, 2000.

[147] W. E. L. Grimson, "A computational theory of visual surface interpolation," A.I. Memo 613, Massachusetts Institute of Technology, June 1981.

[148] R. Perfetti, "Optimization neural network for solving flow problems," *IEEE Transactions on Neural Networks*, vol. 6, pp. 1287–1291, Sept. 1995. Solving Flow Problems with NN's.

[149] A. Bensrhair, M. Bertozzi, A. Broggi, P. Miche, S. Mousset, and G. Toulminet, "A cooperative approach to vision-based vehicle detection," in *IEEE Intelligent Transportation Systems Conference Proceedings*, pp. 207–212, August 2001.

[150] A. Broggi, M. Bertozzi, A. Fascioli, C. L. Bianco, and A. Piazzi, "Visual perception of obstacles and vehicles for platooning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, pp. 164–177, September 2000.

[151] A. Broggi, M. Bertozzi, and A. Fascioli, "Self-calibration of a stereo vision system for automotive applications," in *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, pp. 3698–3703, May 2001.

Appendix A

Appendix A - Project Source

This Appendix will give an overview of various parts of the project, and its source code.

A.1 Definitions

All of the parameters for the pre-processing and matching functions are configurable from defined variables. Some of the parameters are “constant variables”, i.e. they remain constant during a run-though, but change for the next pass based on the output of the previous pass.

The most important is the SEARCHWINSIZE variable which defines how far the network is allowed to ”jump” over poor matches. The run time of the algorithm is proportional to the square of this number, and tests have shown optimal values to be 3 or 4.

The LOOSEST_MATCH Variable defines (in percentage terms) how loose a pixel match can be to be considered for entry into the networking stage. The higher the value, the faster the runtime of the algorithm, but the poorer the algorithm will run in noisy environments.

The NETWORKSIZETHRESHOLD variable defines how large a network should be. Larger networks involve longer run-times. This variable states that if a network grows larger than 400 nodes, the selectiveness of the node chooser is increased by 1% (up to a maximum of 100%).

The MIN_NETWORK_SIZE variable defines the minimum size that a network can be. If a network is created any smaller than this, it is not linked into the system in the flow building step.

```
#define WORST_BLOCK_MATCH 10000
#define INITTHRESH 4
#define SPREAD 0.3   /* Disparities allowed up to 1/3 image.*/
#define SIMTHRESH stereo.linelength*INITTHRESH
#define SEARCHWINSIZE 3
```

```
#define MIN_NETWORK_SIZE 5
#define NETWORKSIZETHRESHOLD 400
#define GOING_UP 0
#define GOING_DOWN 1
#define LOOSEST_MATCH 97
#define MATCH_THRESHOLD 235

int SPREAD_WIDTH=0;     /* Max detected Spread Width */
```

A.2 Data Structures

The STEREO Structure is a holder for all of the image data. It uses the pgm library to load in the images and stores them in the left and right properties

```
struct STEREO
{
  int        linelength;
  int        numlines;
  gray     **left;
  gray     **right;
};

typedef struct STEREO Stereo;
```

The LINK structure defines a link between two nodes. It uses pointers to VERTEX structures to anchor itself between the two nodes. It also has a link to its parent NETWORK. The properties of the link are stored in the fitness and length variables. Storing the link in this way makes it easy to extend its functionality.

```
struct VERTEX;
struct NETWORK;

struct LINK
{
  struct VERTEX* source;
  struct VERTEX* sink;
  int fitness;
  int index;
  struct NETWORK* network;
  double length;
};

typedef struct LINK Link;
```

The VERTEX Structure is the most complex structure in the program. It consists of a link to its parent network in the network variable. This pointer remains null until the vertex is subsumed into a network. The source and sink variables are arrays of link structures which define each of the links upwards (source) and downwards (sink) in the network. The number of sources and sinks are stored in the nsources and nsinks variables. the vertex's location within the 2-d space is keps in the i and j variables. This allows the vertex to be mapped back after reduction. During the network reduction steps, the current "flow value" is kept in the flowval variable, and the length of the path from source to the current vertex is stored in the "flowlen" variable. The "index" and "done" variables are for use when the Queues are being sorted for the Dijkstra reduction step.

```
struct VERTEX
{
  int              index;
  int              i;
  int              j;
  int              nsources;  /* # of vertexes it connects to */
  Link           **source;    /* Possible connections */
  int              nsinks;    /* # of sinks it connects to */
  Link           **sink;      /* their Angles */
  int              flowval;   /* Minimum flow up to here */
  int              flowlen;   /* length of the flow so far */
  struct VERTEX   *parent;    /* Preferred parent! */
  struct NETWORK  *network;   /* for temp storage */
  int              done;
  int              bigstep;
};

typedef struct VERTEX Vertex;
```

As its name suggests, the NETWORK structure holds all the relevant information for a network, and acts as a nucleus for all the flow steps. Each network can have multiple sources and sinks, therefore since dijkstra requires a single source and sink, each network contains a supersource and a supersink vertex which is a ’virtual’ node. All of the sources are evaluated for their ’goodness’, and the initial link between them and their supersource is given a fitness equivalent to this goodness. The same process occurs for the sinks and supersink.

The ’Q’ variable is an array of pointers to vertices. This defines the ’queue’ as used in the Dijsktra algorithm. This queue is constantly sorted and reduced during the algorithm. After the algorithm has completed, the chosen sink and source donate their cartesian coordinates to the x1,y1,x2,y2 variables. This is done so that the later stage of network matching does not need to do unnecessary pointer arithmetic. During building of the networks, the networks are stored in a loose linked list. This is held together with the ’next’ variable. Once the networks are fully built, the linked list is converted into a more structured array for use in the super-network stage.

```
struct NETWORK
{
  int index;     /* Network number in the System */
  Vertex source;/* Network's supersource */
  Vertex sink;   /* Network's supersink */
  int size;      /* Total Size (rough estimate while bldng)*/
  int count;     /* Total size (accurate)*/
  Vertex **Q;    /* Vertex Queue for Relaxation */
  int x1;        /* Top left coordinates for network */
  int y1;
  int x2;        /* bottom right coordinates for network */
  int y2;
  struct NETWORK *next;    /* The Next Network in chain */
};

typedef struct NETWORK Network;
```

Each of the possible paths between two pixels in an nxn grid are stored in a WALK structure. These Walks optimize the link-building step by precomputing the steps needed to traverse the (nxn-1) paths.

```
struct WALK           /* Link value estimator construct */
{
  int pathlen;
  int *pathx;
  int *pathy;
  float *pathval;
};

typedef struct WALK Walk;
```

The SYSTEM structure acts as an anchor point for all of the networks. It can be thought of as the network register. Once a network has been created, it is lodged with the register.

```
struct SYSTEM      /* The entire Network System */
{
  int numnets;
  struct NETWORK **net;
};


typedef struct SYSTEM System;
```

The last stage is the definition of all the globals. This creates all of the information holders for the program.

```
Stereo stereo;         /* holder of all Stereo info */
float lut[255];        /* Lookup table for input conversion */
int *line;
Vertex *vertex;        /* All vertices in the image */
Link *link;            /* Array of All links */
System S;              /* The System */
Walk **walks;          /* nxn array of the walks*/
int vcount;
struct timeval tp,ts;
int debug=0;           /* Set to '1' for debug info*/
int timer=0;           /* Set to '1' for timer info*/
int write_files=0;     /* Write out intermediate working?*/
long s_mallocced = 0;  /* Amount of memory mallocced*/
```

A.3 Cacheing Functions and simple Accellerators

Dealing with raster scan images involves a lot of repeated computation. In order to achieve some performance increases, a few small functions were designed to reduce the number of expensive operations needed to repeated.

A.3.1 Simple Computation Reduction

The first of these is a function, cache_lines() which, given the image sizes, calculates the offset for a particular pixel ina one-dimensional array, given that the images are stored in just such an array internally, and repeated computation of the offset involves excess multiplications.

```
void cache_lines()
{
 int j;

 line = (int*) my_malloc(sizeof(int) * stereo.linelength);

 for (j = 0 ; j < stereo.linelength ; j++)
 {
line[j] = j * stereo.linelength;
 }
}
```

A.3.2 Inter-pixel path fitness cacheing

The cache_walks() function pre-calculates all of the walks from 0,0-¿i,j within the searchspace of nxn. This function incorporates the notion of alisaing, such that if the path travels between two pixels, their values are averaged proportionally to howow much of the line travels through them. In the final system, this equates to path calculation to sub-pixel accuracy.

Precomputing these paths, and storing them in WALK structures, as outlined above saves an enormous amount of computation. The algorithm divides the NxN space into two halves, governed by whether the gradient is less than, or greater than 1. This algorithm is well known in Computer Graphics.

```
void cache_walks()
{
  double  i,j,boxi,boxj,t,di,dj;
  int     cnt,test,numhops,m,n;
  int     pathx[SEARCHWINSIZE*SEARCHWINSIZE];
  int     pathy[SEARCHWINSIZE*SEARCHWINSIZE];
  float pathval[SEARCHWINSIZE*SEARCHWINSIZE];

/* Create the Walks structure as an NxN array */

  walks = (Walk**) my_malloc((SEARCHWINSIZE+1) *
              sizeof(Walk*));
  for (m = 0 ; m < SEARCHWINSIZE+1 ; m++)
    {
     walks[m] = (Walk*) my_malloc((1+SEARCHWINSIZE) *
   sizeof(Walk));
    }

/* Calculate each path from 0,0 -> i,j */
 for (i = 0 ; i < SEARCHWINSIZE; i++)
  for (j = 0 ; j < SEARCHWINSIZE; j++)
      {

/* Increment is governed by the quadrant we are searching. */
        if (i > j)
{ di = 1;   dj = j/i; numhops = i;}
else
{dj = 1; di = i/j; numhops = j;}

boxi=0;
boxj=0;
cnt=0;

/* This travels along the length of the line, which
   is given in 'hops' from pixel to pixel.*/
for (t = 0 ; t < numhops ; t++)
{
boxi += di;
boxj += dj;
test = 0;

/*At each step, calculate whether we're on a pixel,
  or are between two pixels.
  Case 0: directly on a pixel.
  Case 1: between two pixels horizontally.
  Case 2: between two pixels vertically.
  Case 3: Error*/
```

```
if (frac(boxi) >= 0.05) test+=1;
if (frac(boxj) >= 0.05) test+=2;

switch(test)
{
  case 0: pathx[cnt] = (int)boxi;
  pathy[cnt] = (int)boxj;
  pathval[cnt++] = 1;break;

  case 1: pathx[cnt] = (int)boxi;
  pathy[cnt] = (int)boxj;
  pathval[cnt++] = frac(boxi);
  pathx[cnt] = (int)boxi+1;
  pathy[cnt] = (int)boxj;
  pathval[cnt++] = 1.0-frac(boxi); break;

  case 2: pathx[cnt] = (int)boxi;
  pathy[cnt] = (int)boxj;
  pathval[cnt++] = frac(boxj);
  pathx[cnt] = (int)boxi;
  pathy[cnt] = (int)boxj+1;
  pathval[cnt++] = 1.0-frac(boxj); break;

  case 3:  printf("Path Error %lf,%lf\n",boxi,boxj);

}

             }
m = (int)i;
n = (int)j;


/* Create the ''walk'' from pixel to pixel, based on the
  paths generated in the previous step */

walks[m][n].pathx = (int*)my_malloc(cnt * sizeof(int));
walks[m][n].pathy = (int*)my_malloc(cnt * sizeof(int));
walks[m][n].pathval = (float*)my_malloc(cnt * sizeof(float));

for (t = 0 ; t < cnt ; t++)
  {
    walks[m][n].pathx[(int)t] = pathx[(int)t];
    walks[m][n].pathy[(int)t] = pathy[(int)t];
    walks[m][n].pathval[(int)t] = pathval[(int)t];
  }
walks[m][n].pathlen = cnt;
cnt = 0;

        }

}
```

A.3.3 Edge Occlusion Elimination

The cameras do NOT overlap fully. This elim_occlusion() function removes those sections of the images which are invisible to the other camera. This prevents erroneous data from creeping in at the edges of the images, and reduces the amount of data required for matching. The edge occlusion value is passed in as a parameter. In this way, it can change dynamically from frame to frame, if the cameras are movable.

```
void elim_occlusion(gray **crossplane, int occlusion)
{
  int i,j;

  for (i = 1; i < occlusion ; i++)
    {
      for (j = 0 ; j < stereo.linelength ; j++)
{
  crossplane[j][i] = 0;
  crossplane[stereo.linelength-i][j] = 0;
}
    }
}
```

A.4 Network Management Functions

The Networks are managed by a group of simple functions. By keeping all the management into a small set of functions, this keeps the debugging task simple, and makes enhancing the system much easier.

A.4.1 Network Creation

The first function, new_net() creates a new network. That network is returned to the calling function. It checks to make sure there is sufficient memory, and that the creation was successful.

The network is given an 'index' based on the number of nets already registered with the system "s". It then has its source and sink vertices set up.

```
Network *new_net()
{
  Network *N;

  N = (Network*) my_malloc( sizeof( Network ) );
  if (N == NULL)
    {
      fprintf(stderr,"FATAL ERROR:Insufficient Memory \
                      for Alloc\n");
      exit(0);
    }

  N->index = S.numnets++;    /* Set up the net itself */
  N->size = 0;
  N->count = 0;
  N->x1 = stereo.linelength+1;  N->y1 = stereo.linelength+1;
  N->x2 = 0;  N->y2 = 0;

  N->source.index = -1 * N->index;
  N->source.parent = NULL;   /* set up the source */
  N->source.flowval = 1;
  N->source.nsources = 0;
  N->source.nsinks = 0;
  N->source.source = NULL;
  N->source.sink = NULL;
  N->source.done = 1;
  N->source.flowlen = 1;
  N->source.i = -99;
  N->source.j = -99;


  N->sink.index = -1 * N->index;  /* set up the sink */
  N->sink.parent = NULL;
  N->sink.flowval = 0;
  N->sink.nsources = 0;
  N->sink.nsinks = 0;
  N->sink.source = NULL;
  N->sink.sink = NULL;
  N->sink.flowlen = 0;
  N->sink.done = 1;
  N->sink.i = -99;
  N->sink.j = -99;
  return N;
}
```

A.4.2 Network Sink Addition

The add_sink() function adds a potential sink vertex to the network. It is passed the network to be grafted onto, and the Link structure which links to the sink vertex.

The function first checks whether the sink array should be enlarged. The typical number of sources to a vertex is 4 or fewer, therefore, if the number of sources grows larger than four, it is enlarged. This does not happen with a 3x3 search window.

Then, using pointers, it attaches the link to the sink vertex list of the network.

```
void add_sink(Network *N, Link *L)
{
 if (N->sink.nsources % 5 == 0)
   {
     N->sink.source = (Link**) realloc (N->sink.source,
      (N->sink.nsources+6)*sizeof(Link*));
     if (N->sink.source == NULL)
       {
 fprintf(stderr,"FATAL ERROR:Insufficient Memory \
         for Alloc\n");
 exit(0);
       }
   }
 L->sink = &N->sink;                    /* connect the link */
 N->sink.source[N->sink.nsources++] = L;/* connect the sink */
}
```

A.4.3 Network Source Addition

The add_source() function works in exactly the same way as the add_sink() function, however it grafts a source onto the beginning of a network.

```
void add_source(Network *N, Link *L)
{

/*Is the array grown so large that we need to enlarge it?*/
  if(N->source.nsinks % 5 == 0)
    {
      N->source.sink = (Link**)realloc(N->source.sink,
     (N->source.nsinks+6) * sizeof(Link*));
     if (N->source.sink == NULL)
       {
 fprintf(stderr,"FATAL ERROR:Insufficient \
  Memory for Alloc\n");
 exit(0);
       }
    }

/* connect the link */
  L->source = &N->source;
/* connect the source */
  N->source.sink[N->source.nsinks++] = L;


}
```

A.4.4 Setting the Network Inter-node Link Weights

Once the networks have been created from their various vertices, and the inter-vertex links have been established, the edge weights must be calculated. These weights are determined by the values of the pixels which lie on a line between the various vertices. For this, the pre-calculated paths are employed to calculate the edge weights.

The function is recursive, and is started by providing it with the source vertex. It then traverses across the network in a node-to-node fashion until it has completely traversed the whole network, setting up the link weights.

```
void trace_network(Link *curlink, Network *N,
                   gray** crossplane, int Direction)
{
  int i,j,si,sj;
  int ii,jj,n;
  float t,tt,tbd;
  Vertex *next;

/* Add current link to the network */
  curlink->network = N;
  if ((Direction == GOING_DOWN) && (curlink->sink == NULL))
    {
      add_sink(N,curlink);
      return;  /* Bottom of trace */
    }

  if ((Direction == GOING_UP) && (curlink->source == NULL))
    {
      add_source(N,curlink);
      return;  /* top of Trace */
    }
  if(debug)fprintf(stderr,"Calculating walkval for \
                    link %d\n",curlink->index);


  if ((curlink->source != NULL) &&
      (curlink->source->index != N->source.index))
    {
      si = curlink->source->i;
      i = curlink->sink->i - si;
      sj = curlink->source->j;
      j = curlink->sink->j - sj;
/* Amount to divide the total by at the end */
      tbd = 1.0;
/* Start off with the first value */
      t = crossplane[sj][si];

/* Calculate the link's flowvalue for relaxation*/
      for (n = 0 ; n < walks[i][j].pathlen ; n++)
{
  ii = si + walks[i][j].pathx[n];
  jj = sj + walks[i][j].pathy[n];
  tt = (float)(crossplane[jj][ii]) *
                walks[i][j].pathval[n];
  tbd += walks[i][j].pathval[n];
  t += tt;
}
```

```
      t = (t / tbd);
    }
  else{t = 0; i = 0; j = 0;}

 curlink->fitness = (int)t;

/* Keep traversing the net in the right direction!*/
 if (Direction == GOING_DOWN)
   {
     next = curlink->sink;
   }
 else
   {
     next = curlink->source;

   }
  next->network = N;

  if (curlink->sink == NULL)
  {
  fprintf(stderr,"Treversed too far!\n");
  }
  if (curlink->source == NULL)
  {
fprintf(stderr,"Traversed too far!\n");
  }

/* now, do down branches */
  for(i = 0 ; i < next->nsinks ; i++)
    {
      if ((next->sink[i])->network != N)
{
  N->size++;
  trace_network(next->sink[i],N,crossplane,GOING_DOWN);
}
    }

/* Now, do up branches */
  for(i = 0 ; i < next->nsources ; i++)
    {
      if ((next->source[i])->network != N)
{
  N->size++;
  trace_network(next->source[i],N,crossplane,GOING_UP);
}
    }
}
```

A.4.5 Network Source and Sink Link-Weight Setting

Once the network has been created, the sinks and sources are all links to the super source and super sink. Their edge weights to these virtual nodes are governed by their location relative to the network, and are not set by the trace_network() function since their edge-weights are not governed by pixel values, but by pixel location. This function is provided specifically to set the weights of these virtual links.

As a by-product of performing this on the networks, the extent variables x1,y1,x2,y2 are set up, which define the network's horizontal and vertical dimensions.

The networks which touch the edges of the correlation plane typically do not conform to the same metric as "inner-networks". This is because objects at the image edges are often quite close compared to objects in the middle of the scene (see the boxes and corridor datasets). For this reason, the network is truncated. If the image was larger, then the network would indeed end at the top-left-most and bottom-right-most vertex, however the edge of the plane cuts the network short, so the final edge weight is calculated from the vertex fitness value. This does not work on inner networks, and often breaks on edge-networks, but is does at least work.

```
void fix_superlinks(Network *N)
{
 int i,hi,hj,li,lj;
 if(debug)pr_time("fs_start");

/* check to make sure the network is valid*/
 if ((N->source.nsinks == 0) || (N->sink.nsources == 0))
   {
     fprintf(stderr," Empty or erroneous network %d.\n", \
                        N->index);
     return;
   }

 * first, find the top left most source */
  li = N->source.sink[0]->sink->i;
  lj = N->source.sink[0]->sink->j;
  if (N->source.nsinks > 1)
   {
     for (i = 1 ; i < N->source.nsinks ; i++)
        {
 if (N->source.sink[i]->sink->j < lj)
   {
    lj = N->source.sink[i]->sink->j;
  }
if (N->source.sink[i]->sink->i < li)
  {
    li = N->source.sink[i]->sink->i;
  }
      }
  }

 N->x1 = li;
 N->y1 = lj;



/* now find the bottom right-most sink */
 hi = N->sink.source[0]->source->i;
 hj = N->sink.source[0]->source->j;
 if (N->sink.nsources > 1)
  {
    for (i = 1 ; i < N->sink.nsources ; i++)
      {
if (N->sink.source[i]->source->j > hj)
  {
    hj = N->sink.source[i]->source->j;
  }
if (N->sink.source[i]->source->i > hi)
  {
    hi = N->sink.source[i]->source->i;
  }
      }
  }


 N->x2 = hi;
 N->y2 = hj;

/* Now li, lj and hi, hj describe the fullest extents
   of the Net. Now we use these values to calculate
   the values of the superlinks, UNLESS it's the first
   or last block on the screen*/

 for ( i = 0 ; i < N->source.nsinks ; i++)
  {
    if((N->x2 < stereo.linelength-5) &&
       (N->y2 < stereo.linelength-5))
      {
hi = abs(N->source.sink[i]->sink->i-N->x1);
```

```
 hj = abs(N->source.sink[i]->sink->j-N->y1);
 N->source.sink[i]->fitness = 255-(hi + hj);
       }
     else
       {
 N->source.sink[i]->fitness = 254;
       }
   }


  for ( i = 0 ; i < N->sink.nsources ; i++)
   {
     if ((N->x1 > 5) && (N->y1 > 5))
       {
 hi = abs(N->sink.source[i]->source->i-N->x2);
 hj = abs(N->sink.source[i]->source->j-N->y2);
 N->sink.source[i]->fitness = 255 - (hi + hj);
       }
     else
       {
 N->sink.source[i]->fitness = 254;
       }
   }
  if(debug)fprintf(stderr,"Network %d ranges from \
%d,%d -> %d,%d\n" \
                  ,N->index,N->x1,N->y1,N->x2,N->y2);

  pr_time("fs_end");

}
```

A.4.6 Per-Network Dijkstra Queue Building

The build_sets() function takes all of the vertices have been assigned to a NETWORK structure, and builds the Dijkstra Queue sets out of the for each Network, in preparation of the reduction step.

```
void build_sets()
{
  int i=0;
  Network *N;

  pr_time("bs_start");
/* For each vertex */
  for (i = 0 ; i < vcount ; i++)
    {

/* Get the network it's in */
      N = vertex[i].network;
      if (N == NULL)
      {
fprintf(stderr,"WARNING: A Vertex (%d) is not \
assigned to a Network\n",i);
      }
      else
      {

/* If insufficient Size! */
        if (N->count > N->size-2)
        {
/* Enlarge the array */
          N->size = (N->size+1) * 2;
          N->Q = (Vertex**)realloc(N->Q,
                        N->size*sizeof(Vertex*));
        }

/* Stick it in the Queue */
        N->Q[N->count++] = &vertex[i];
      }
    }
 pr_time("bs_end");

}
```

A.4.7 Outlier Vertex Removal

The strip_outliers() function finds all those vertixes which do not have any neighbours within their allowable SEARCHWINDOWxSEARCHWINDOW

connection-zone. This vastly reduces the number of 1-node networks created by noise, or accidental matches. It also removes many 'impossible' vertices, i.e. those which can never be connected to, even though they are in proximity to other networks. This optimizes the following steps to a huge extent.

The 'gatearray' parameter holds an nxn matrix of the same size as the correlation plane. This array contains a set of boolean values based upon whether the pixel at that point is a 'gate', i.e. whether the network path could pass through that point. Initially, all possible vertices are marked as 'gates', and then the gate array is reduced by steps such as this one until only the maximum likelihood gates are remaining.

```
int strip_outliers(int** gatearray, int numgates)
{
 int i,j,ii,jj,ok = 0,sok=0;
 pr_time("so_start");

/* Search the whole space*/

 for (j = 0 ; j < stereo.linelength ; j++)
  for (i = 0 ; i < stereo.linelength ; i++)
   {
ok = 0;
sok = 0;

/* If that pixel is identified as being 'interesting*/

if (gatearray[j][i] != 0)
 {

/* Search within its Search Window for vertices \
   that can connect to it */

  for (jj = j-(SEARCHWINSIZE) ;
(jj < j) && (ok == 0) ; jj++)
   for (ii = i-(SEARCHWINSIZE) ;
 (ii < i) && (ok == 0) ; ii++)
{
 if (jj < 0) {jj = 0;}
 if (ii < 0) {ii = 0;}

/* If one is found, remember that the vertex is 'ok'*/

 if (gatearray[jj][ii] != 0){ok = 1;}
}
  sok = ok;

/* Seach within its Search Window for Vertices \
   that it can connect to */
   for (jj = j+1 ; (jj < j+SEARCHWINSIZE) &&
                           (jj < stereo.linelength) &&
   (ok == sok)  ; jj++)
            for (ii = i+1 ; (ii < i+SEARCHWINSIZE) &&
                            (ii < stereo.linelength) &&
    (ok == sok) ; ii++)
{
  if (jj > stereo.linelength) {jj = stereo.linelength;}
  if (ii > stereo.linelength) {ii = stereo.linelength;}

/* Once again, if a connector is found, remember \
   that the vertex is ok*/

  if (gatearray[jj][ii] != 0){ok++;}
}

/* If the vertex is isolated, remove it from the \
   array of allowed vertices*/

    if (ok == 0)
      {
gatearray[j][i] = 0;
numgates--;
      }
  }
      }
  pr_time("so_end");

/* Return the new number of 'gates', i.e.
   vertices through which the path can flow*/

  return numgates;
}
```

A.5 The Dijkstra Flow-Reduction Algorithm

The heart of the flow-reduction is the Dijkstra Shortest Path algorithm.

The relax_network() function is a C version of the Dykstra Shortest-Path Algorithm. It has been modified to take into consideration the continuity and smoothness constraints.. There has been a large chunk of testing code bolted onto its front.. Hopefully this will catch any errors that would Throw the relaxer off.

```
void relax_network(Network *N)
{
  Vertex *V,*V2 , *Vtmp;
  int i,wij,bestval,best,end,j;
  int numleft;    /* # of non-NULL Vertices in Q*/
  int upds=0;
  int tmp = 0,origcount=0,flowinc=0;
  double m1,m2;

   N->source.flowval = 1;    /* Initialise the Flow Values */
   if (debug)fprintf(stderr,"Adding vertex %d \
to network %d\n", \
                       N->sink.index,N->index);
   N->Q[N->count++] = &(N->sink); /* Add the Sink vertex;*/

  tmp = 0;

/* Start with the initial supersource vertex */
  V = &(N->source);
  end = N->count-1;

  best = 0;
  do
    {

/* Having picked out the best, relax
 * all its children */

      for (i = 0 ; i < V->nsinks ; i++)
{
  V2 = V->sink[i]->sink;

  if (V2 != NULL)             /* Not the sink links! */
    {

/* Calculate the 'wij' value, i.e. the weight
 * of the edge between the sink and current
 * vertex*/

      wij = V->flowval + V->sink[i]->fitness;

/* Next, calculate the cartesian distance for
the flow's length*/

      m1 = (V2->j - V->j)*(V2->j - V->j);
      m2 = (V2->i - V->i)*(V2->i - V->i);
      flowinc = sqrt(m1+m2);

/*Should this source become the new parent?/*

      if (((wij) > (V2->flowval)) ||
         (((wij) == (V2->flowval)) &&
 (V->flowlen+flowinc < V2->flowlen)))
     {
/* Update the flow value*/
       V2->flowval = wij;
/* change the parent */
       V2->parent = V;
/* Increase the flow length accordingly*/
       V2->flowlen = V->flowlen+flowinc;
     }
    }
}

            bestval = -2;
    best = 0;
    for(i=0 ; i <= end ; i++)
      {
if ((N->Q[i]->flowval > bestval) ||
    ((N->Q[i]->flowval == bestval) &&
     (N->Q[i]->flowlen < N->Q[best]->flowlen)))
  {
    best = i; bestval = N->Q[i]->flowval;
  }
      }

/* Take the best vertex in the queue*/
    V = N->Q[best];
/* Move this one to the end */
    N->Q[best] = N->Q[end];
/* Reduce the queue length so that the
   one we just moved isn't considered in
   the next round*/
    N->Q[end--] = V;


    }
  while(end >= 0);
/* While there's stuff still in the Queue */


/* Check to make sure it all went smoothly */
  if (N->sink.parent == NULL)
  {
       fprintf(stderr,"RELAX ERROR: Sink is still \
        unresolved for this net (mod flag %d) %d\n",
        N->index,upds);
return;
  }

}
```

A.5.1 Flow Building

This function is the longest in the program. It takes the raw pixels in the correlation plane, and constructs the list of all vertices from it.

It first builds the "gate array". As described above, the gate array is just an nxn matrix, with the same dimensions as the correlation plane. Each cell (i,j) in the gate array denotes whether the pixel (i,j) in the correlation plane is viable for matching.

After creating all the flows, it then makes the networks, fixes all the link weights, fixes the superlink weights, and then at the very end runs the Dijkstra shortest path algorithm on them.

This function forms the heart of the Network-related section of the program.

```
void buildflows(gray **crossplane,
int** gatearray,
int numgates)
{
  int i,j;
  int lj,comp;
  int vert, vert2, nv=0;
  int nlinks = 0;
  Network first;
  Network *tmpnet;

  first.next = NULL;

  pr_time("bf_start");

/* Link is set up with the ABSOLUTE MAXIMUM number
 * of possible links.this will probably never occur. */
```

```
  link = (Link*) my_malloc (2 * numgates *
 SEARCHWINSIZE*SEARCHWINSIZE *
 sizeof(struct LINK));
  if (link == NULL)
    {
      fprintf(stderr,"FATAL ERROR: Insufficient \
      memory for Alloc");
      exit(0);
    }

/* Create the list of all vertices */

  vertex = (Vertex*) my_malloc (1+numgates * 2 *
        sizeof(struct VERTEX));

  if (vertex == NULL)
    {
      fprintf(stderr,"FATAL ERROR: Insufficient \
      memory for Alloc");
      exit(0);
    }

/* Creating all the vertices based on
   the gray-level gate array */
  for (j = 0 ; j < stereo.linelength-1 ; j++)
    {
      lj = line[j];
      for ( i = (j<SPREAD_WIDTH)?0:j-(SPREAD_WIDTH) ;
 i < stereo.linelength && (i < j+SPREAD_WIDTH);
 i++)
{
  if (gatearray[j][i] == 1)
    {
      vertex[vcount].i = i;  vertex[vcount].j = j;
      vertex[vcount].nsources = 0;
      vertex[vcount].nsinks = 0;
      vertex[vcount].sink = (Link**) my_malloc
              (SEARCHWINSIZE*SEARCHWINSIZE * sizeof(Link*));
      vertex[vcount].source = (Link**)my_malloc
        (SEARCHWINSIZE*SEARCHWINSIZE * sizeof(Link*));
             vertex[vcount].flowval = 0;
      vertex[vcount].index = vcount;
      vertex[vcount].network = NULL;
      vertex[vcount].flowlen = 0;
      vertex[vcount].done = 0;
      vertex[vcount].bigstep = 0;
      vertex[vcount++].parent = NULL;
    }
}
    }

   pr_time("bf_setupend");

/* go through each vertex, creating join and sink links */

  for (vert = 0 ; vert < vcount ; vert++)
    {
      lj = line[vertex[vert].j];

/* Number of links for this vertex = 0; */
      nv = 0;
      comp=0;

/* Find all follow-on vertices from the sorted list
   of all vertices */
      for(vert2 = vert+1 ;
  ((vert2<vcount) && (vertex[vert2].j<=
      vertex[vert].j+SEARCHWINSIZE));
  vert2++)
       {

/* Is this vertex connectable to the current
   vertex under inspection? If so, link them*/
  if((vertex[vert].j < vertex[vert2].j) &&
     (vertex[vert].i < vertex[vert2].i) &&
     (vertex[vert2].j - vertex[vert].j < SEARCHWINSIZE) &&
     (vertex[vert2].i - vertex[vert].i < SEARCHWINSIZE))
    {

     vertex[vert].sink[vertex[vert].nsinks++] =
&link[nlinks];
     vertex[vert2].source[vertex[vert2].nsources++] =
&link[nlinks];
     link[nlinks].source = &vertex[vert];
       link[nlinks].sink = &vertex[vert2];
     link[nlinks].index = nlinks;
     link[nlinks].fitness = 0;
     link[nlinks++].network = NULL;
    }
}

    }
```

```
 for (vert2 = 0 ; vert2 < vcount ; vert2++)
  {
/* Create the Source Links */
   if(vertex[vert2].nsources == 0)
{
 vertex[vert2].source[vertex[vert2].nsources++] =
   &link[nlinks];
 link[nlinks].source = NULL;
 link[nlinks].sink = &vertex[vert2];
 link[nlinks].index = nlinks;
 link[nlinks].fitness = 0;
 link[nlinks++].network = NULL;
}

   if (vertex[vert2].nsinks == 0)
{
/* Create a Sink Link */
 vertex[vert2].sink[vertex[vert2].nsinks++] =
   &link[nlinks];
 link[nlinks].network = NULL;
 link[nlinks].source = &vertex[vert2];
 link[nlinks].sink = NULL;
 link[nlinks].index = nlinks;
 link[nlinks].fitness = 0;
 link[nlinks++].network=NULL;
}
 }


 pr_time("bf_linksdone");

 S.numnets = 0;

 for (i = 0 ; i < nlinks ; i++)
  {
/* Find the source links! */
   if ((link[i].source == NULL) &&
 (link[i].network == NULL))
{

 tmpnet = new_net();
 add_source(tmpnet,&link[i]);   /* Add first source */

 trace_network(&link[i],tmpnet,crossplane,GOING_DOWN);
 tmpnet->Q = (Vertex**) my_malloc (tmpnet->size *
sizeof(Vertex*));
/* Link Tmpnet onto the Network List if it has more than a
   threshold number of nodes*/
  if (tmpnet->size>MIN_NETWORK_SIZE){
  tmpnet->next = first.next;
  first.next = tmpnet;}

}
    }

  pr_time("bf_netsdone");
  build_sets();
  pr_time("bf_setsbuilt");

/* Stick all the Networks, in order, into the System   */

  S.net = (Network**)realloc(S.net, \
                     S.numnets * sizeof(Network*));
  for (i = 0 ; i < S.numnets ; S.net[i++] = NULL);

  tmpnet = first.next;


tmpnet = first.next;

  pr_time("bf_links_fixed");
  i = 0;

/* Now fix all the superlinks, and then run the network
   relaxation.  This actually performs the dijkstra
   shortest path algorithm*/

  while(tmpnet != NULL)
    {
      fix_superlinks(tmpnet);
      relax_network(tmpnet);
      S.net[i] = tmpnet;
      tmpnet = tmpnet->next;
      i++;
    }
  S.numnets = i;
  pr_time("bf_netsinserted ");

}
```